Introduction to Divide and Conquer Approach Searching Binary Search Sorting, Merge Sort

Week-01, Lecture-02

Course Code: CSE221

Course Title: Algorithms

Program: B.Sc. in CSE

Course Teacher: Tanzina Afroz Rimi

Designation: Lecturer

Email: tanzinaafroz.cse@diu.edu.bd

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

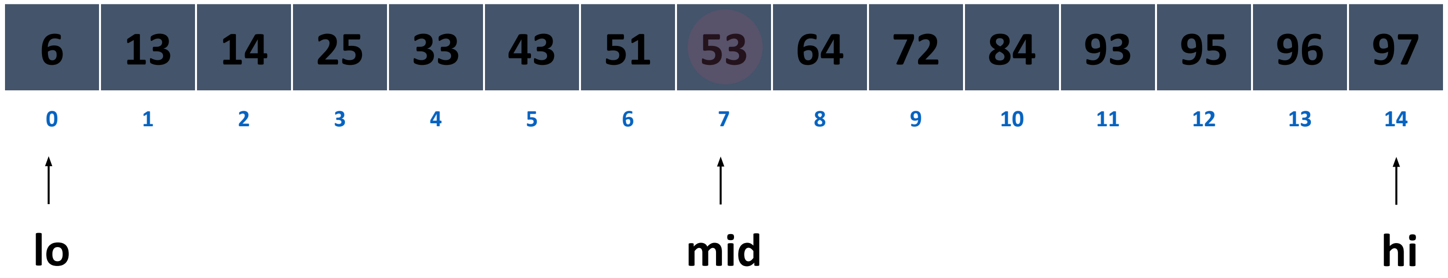
6	**13**	**14**	**25**	**33**	**43**	**51**	**53**	**64**	**72**	**84**	**93**	**95**	**96**	**97**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
lo														**hi**

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.



- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

lo **hi**

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
				↑		↑								
				lo		**hi**								

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

6	13	14	25	**33**	**43**	**51**	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
				lo	**mid**	**hi**								

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

6	13	14	25	**33**	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

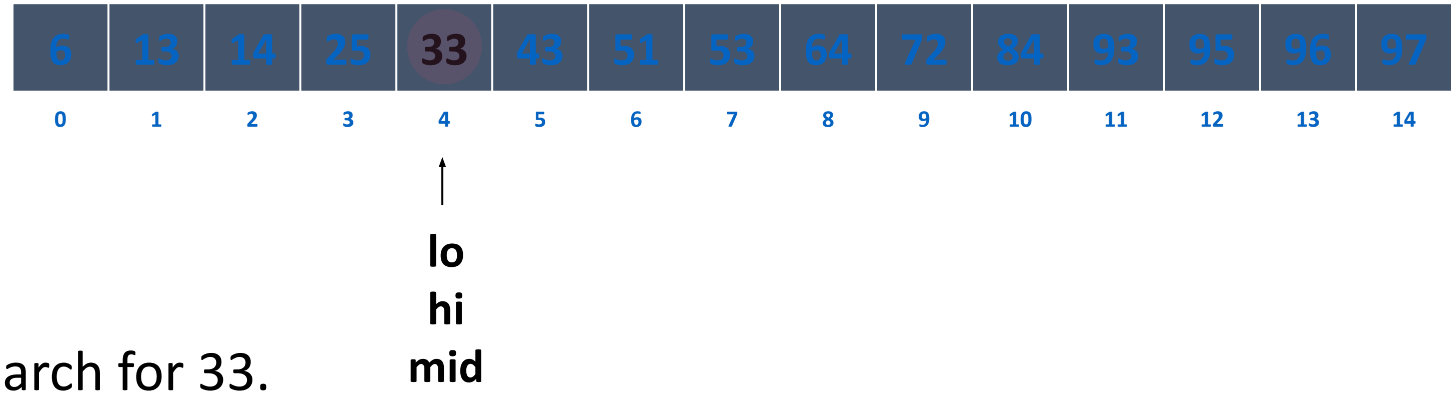
lo
hi

- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.



- Ex. Binary search for 33.

Binary Search

- Binary search. Given `value` and sorted array `a[]`, find index `i` such that `a[i]` = `value`, or report that no such index exists.

- Invariant. Algorithm maintains `a[lo]` ≤ `value` ≤ `a[hi]`.

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

lo
hi
mid

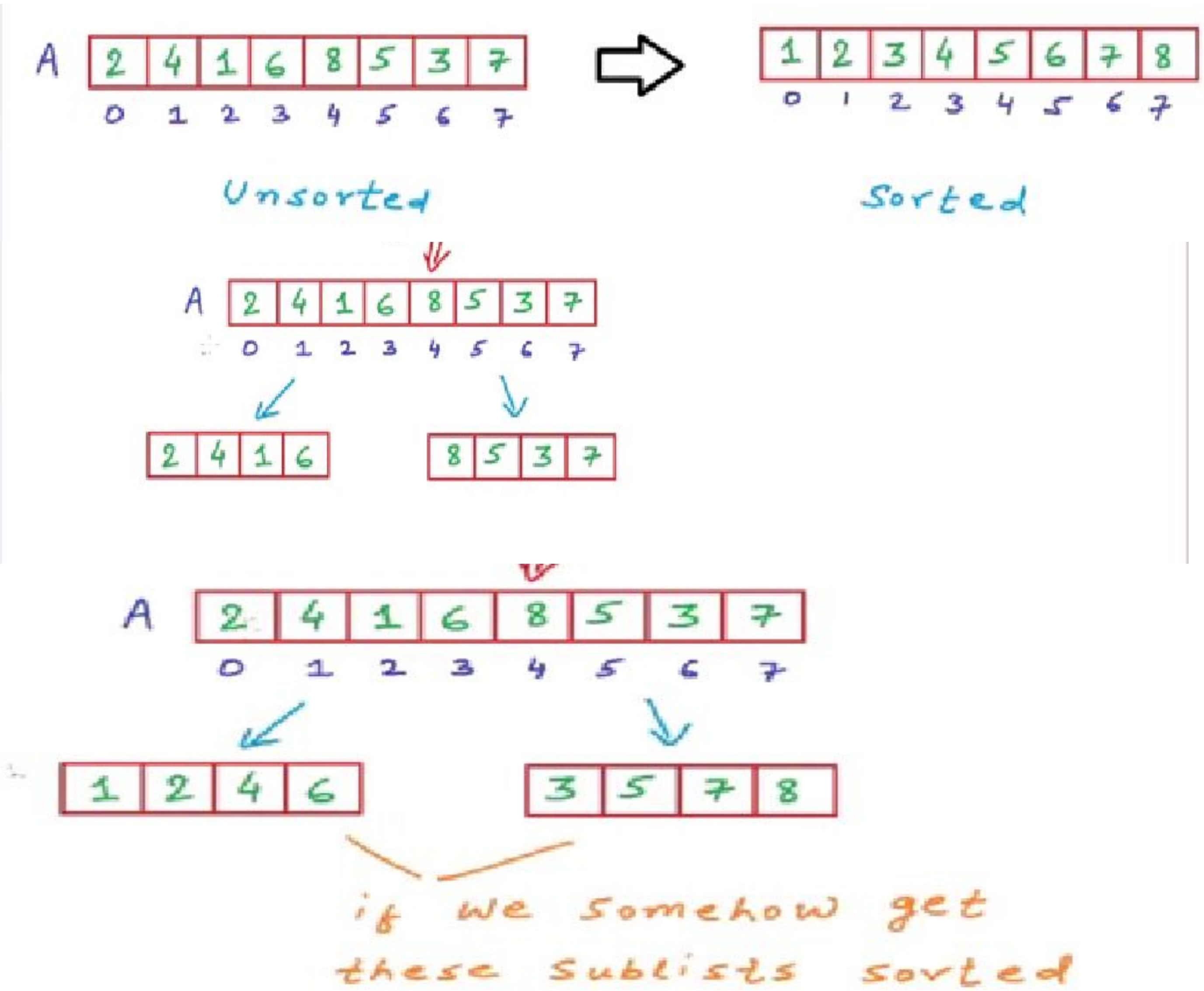
- Ex. Binary search for 33.

Merge Sort

Divide & Conqure

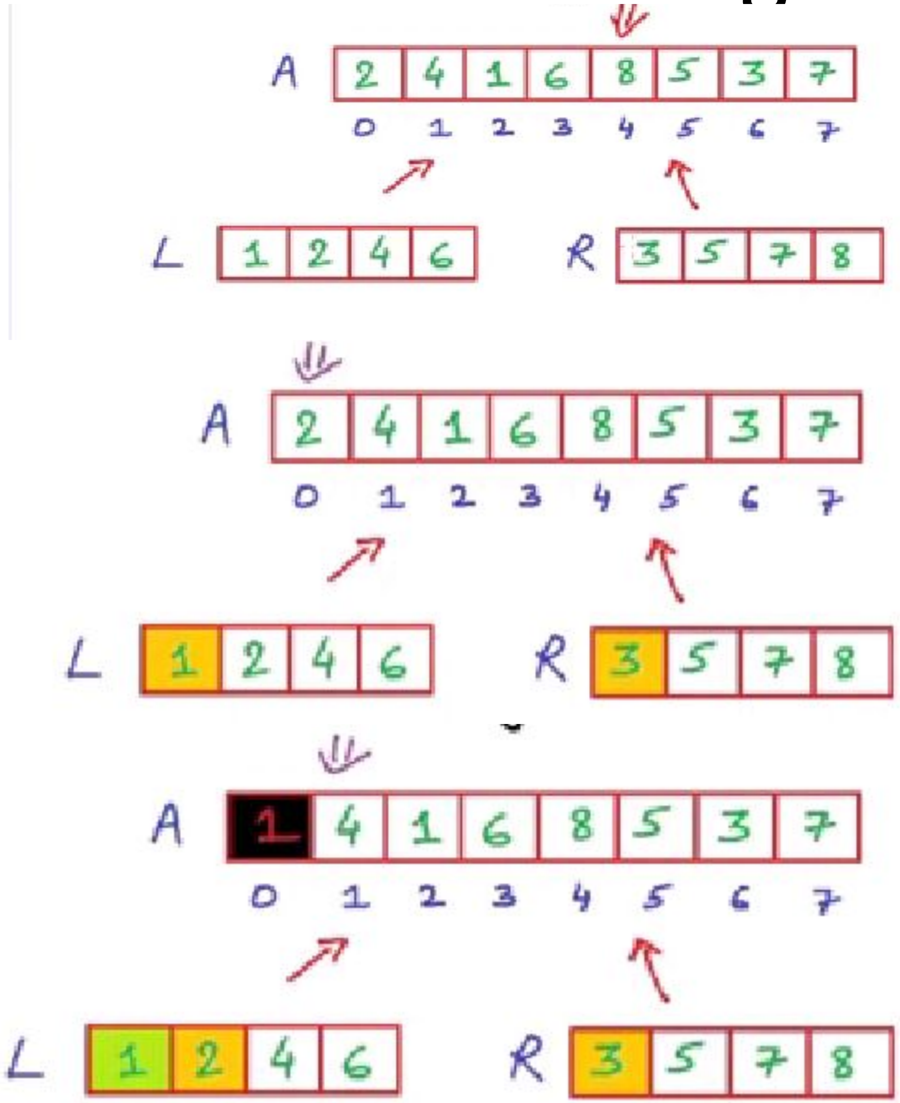
- The divide-and-conquer paradigm involves three steps at each level of the recursion:
  - **Divide** the problem into a number of subproblems.
  - **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
  - **Combine** the solutions to the subproblems into the solution for the original problem.
- The ***merge sort*** algorithm closely follows the divide-and-conquer paradigm. Intuitively, it operates as follows.
  - **Divide:** Divide the n-element sequence to be sorted into two subsequences of $n/2$ elements each.
  - **Conquer:** Sort the two subsequences recursively using merge sort.
  - **Combine:** Merge the two sorted subsequences to produce the sorted answer.

Merge Sort



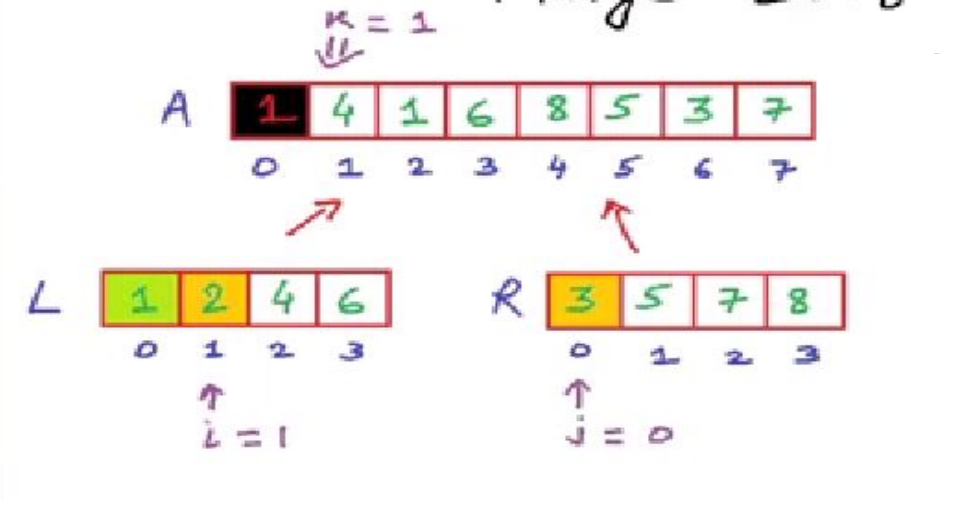
A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
Unsorted
1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7
Sorted
A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
2 4 1 6
8 5 3 7
A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
1 2 4 6
3 5 7 8
if we somehow get
these sublists sorted

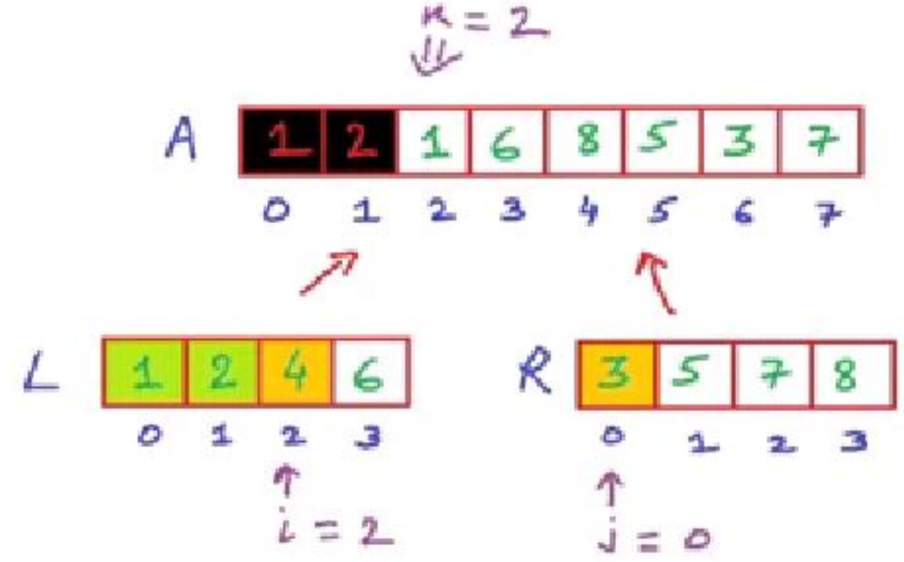
Merge..



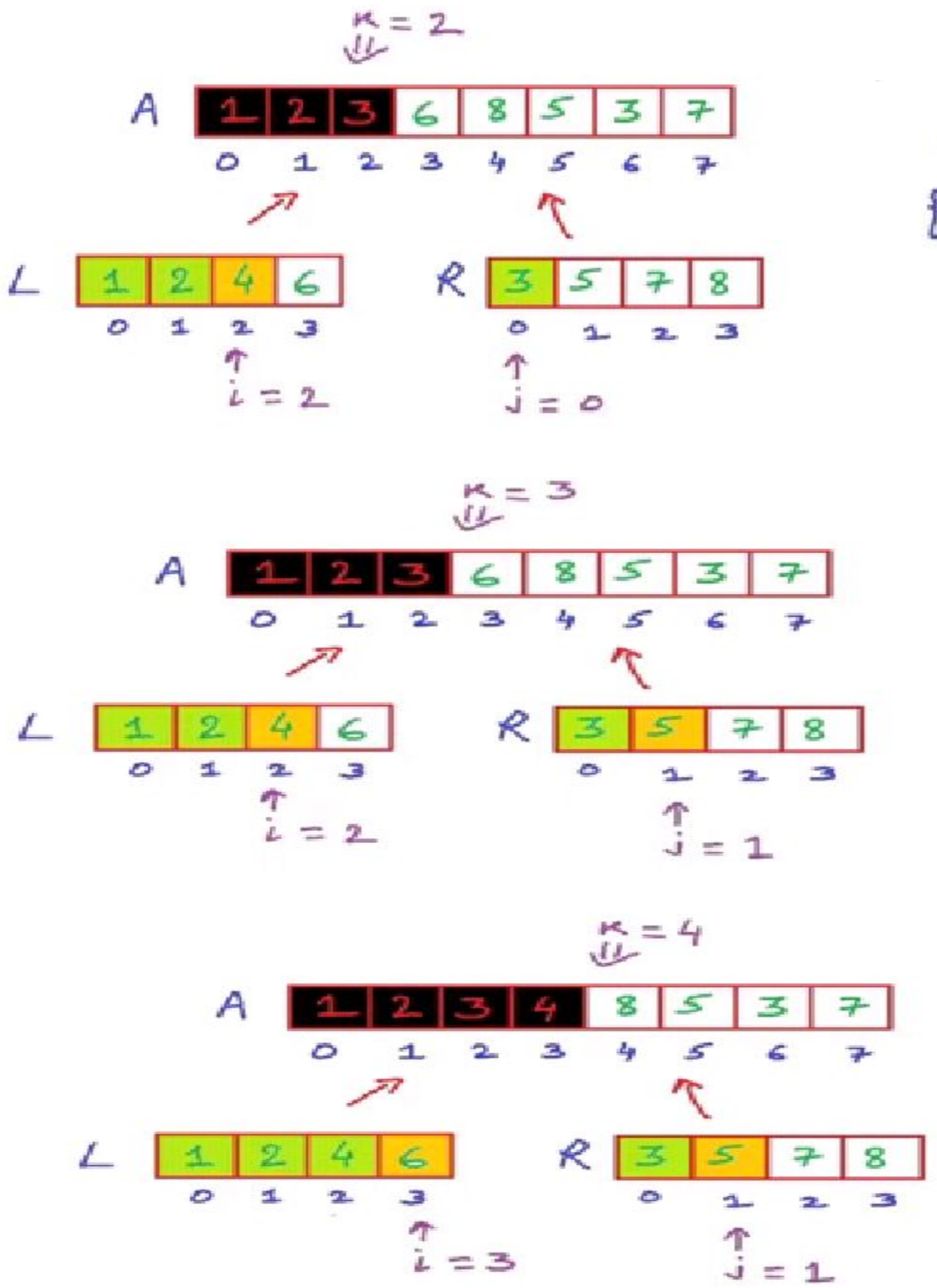
A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
L
1 2 4 6
R
3 5 7 8
A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
L
1 2 4 6
R
3 5 7 8
A
1 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
L
1 2 4 6
R
3 5 7 8

Merge Sort

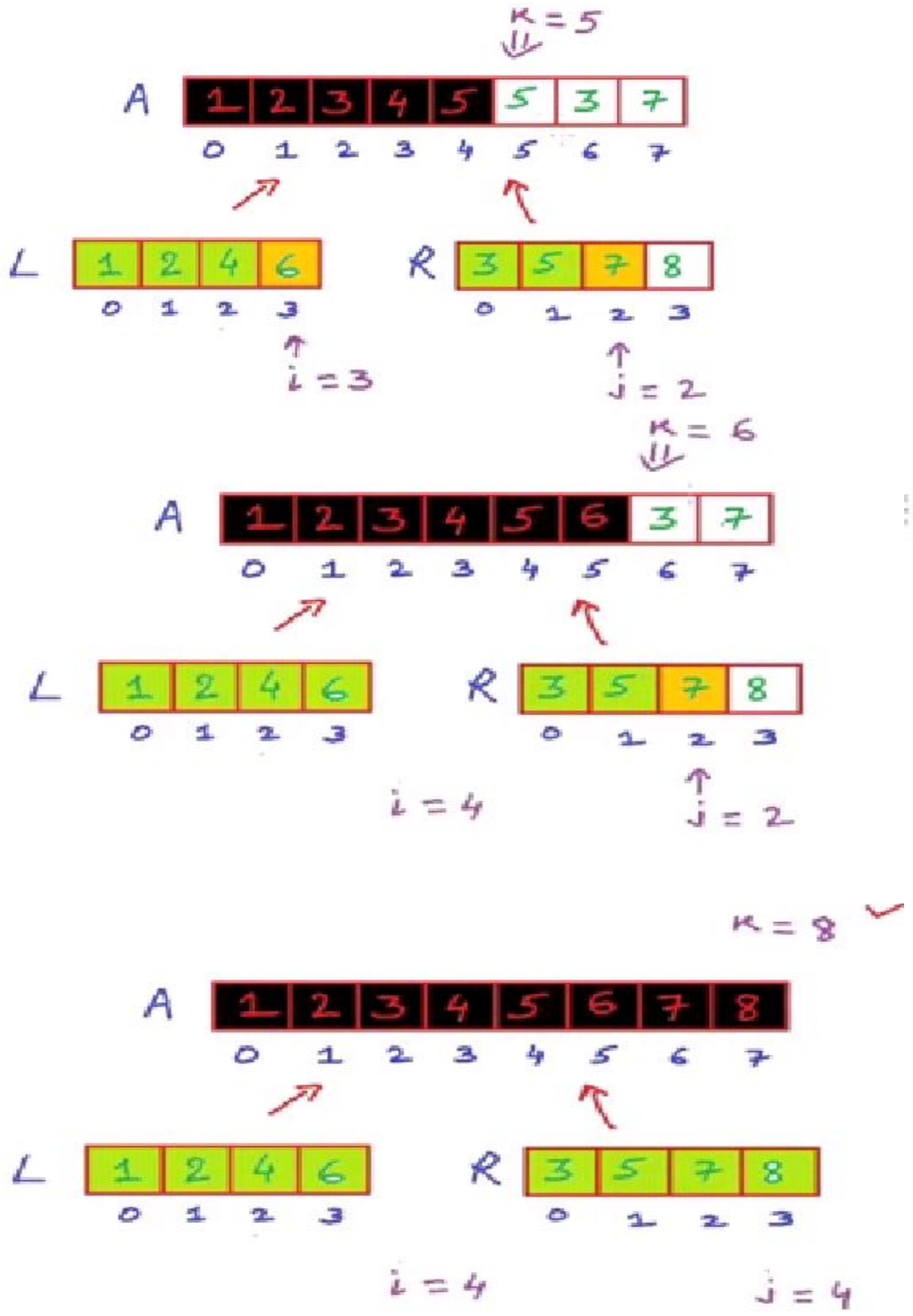





```
Merge(L, R, A)
{
   nL ← length(L)
   nR ← length(R)
   i ← j ← k ← 0
  while(i < nL && j < nR)
  {
    if( L[i] <= R[j])
    {
       A[k] ← L[i]
       i ← i+1
    }
    else
    {
       A[k] ← R[j]
       j ← j+1
    }
    k ← k+1
  }
```

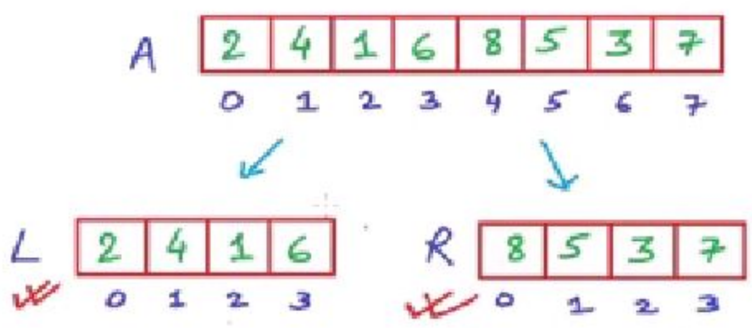


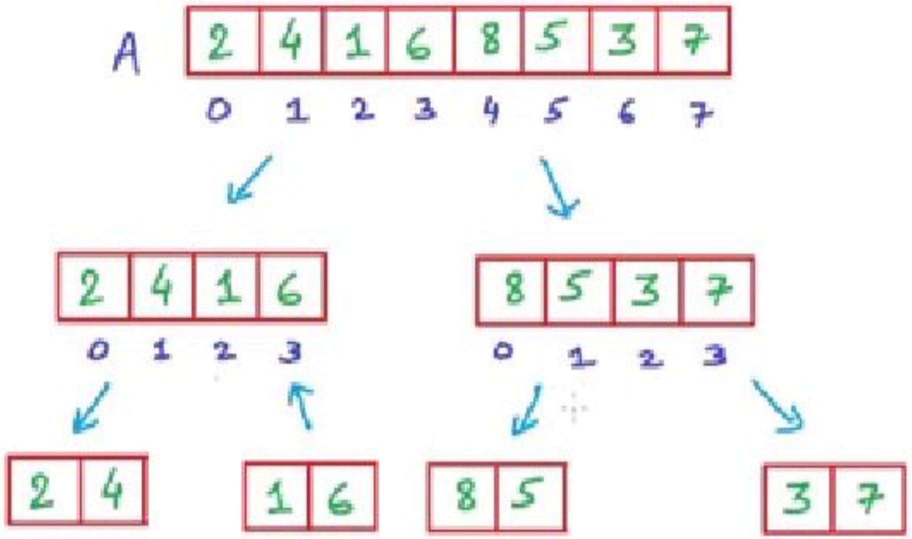

```
Merge(L, R, A)
{
  nL ← length(L)
  nR ← length(R)
  i ← j ← k ← 0
  while(i < nL && j < nR)
  {
    if( L[i] <= R[j])
    {
      A[k] ← L[i]
      i ← i+1
    }
    else
    {
      A[k] ← R[j]
      j ← j+1
    }
    k ← k+1
}
```

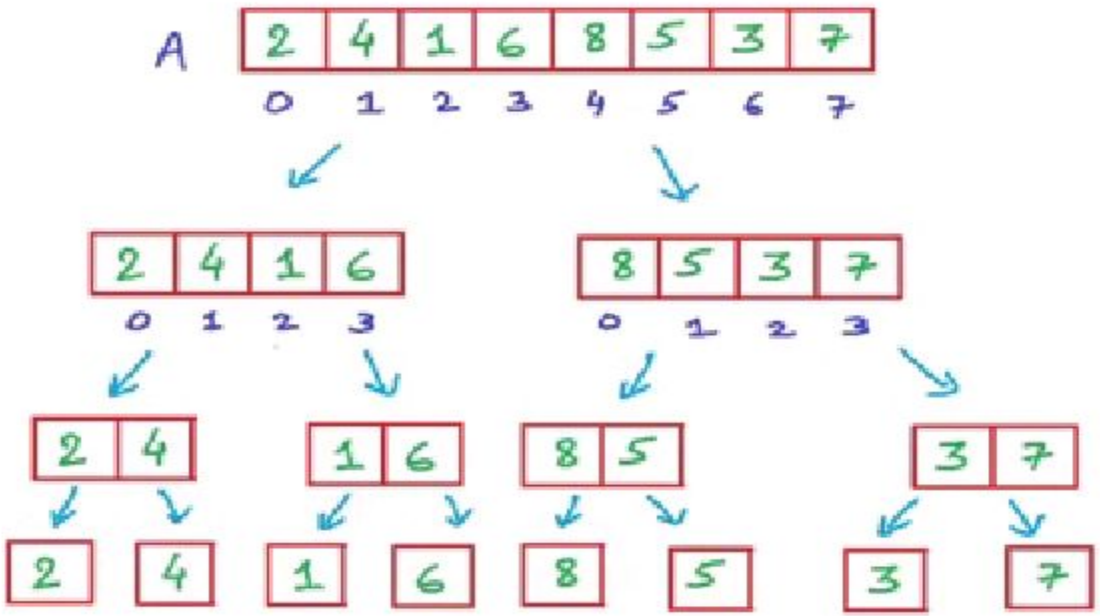


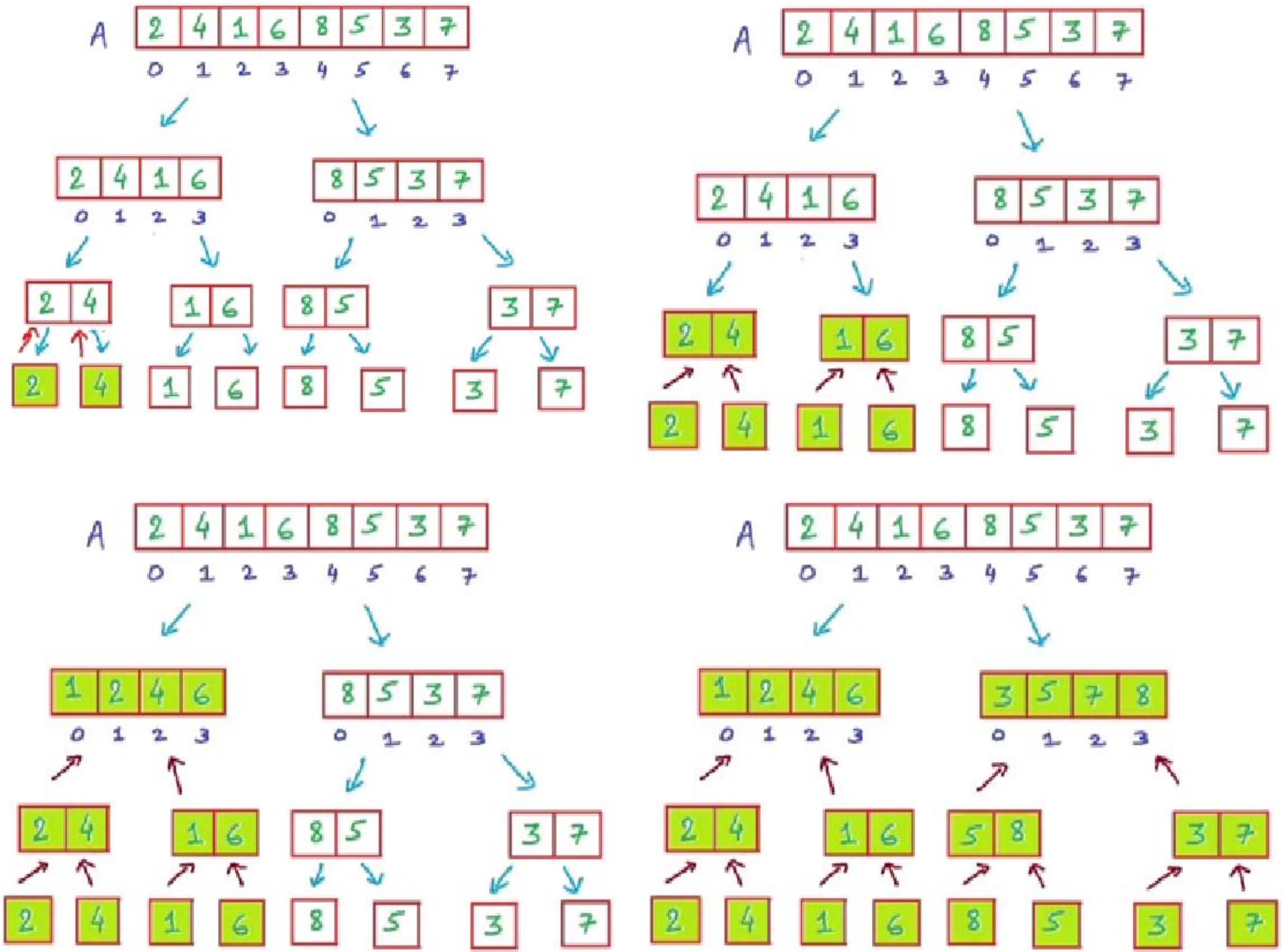

```
Merge(L, R, A)
{
  nL ← length(L)
  nR ← length(R)
  i ← j ← k ← 0

  while(i < nL && j < nR)
  {
    if( L[i] <= R[j])
    {
      A[k] ← L[i]; i ← i+1
    }
    else
    {
      A[k] ← R[j]; j ← j+1
    }
    k ← k+1
  }
  while (i < nL)
  { A[k] ← L[i]; i←i+1; k←k+1;
  }
  while(j < nR)
  { A[k] ← R[j]; j←j+1; k←k+1;
  }
```

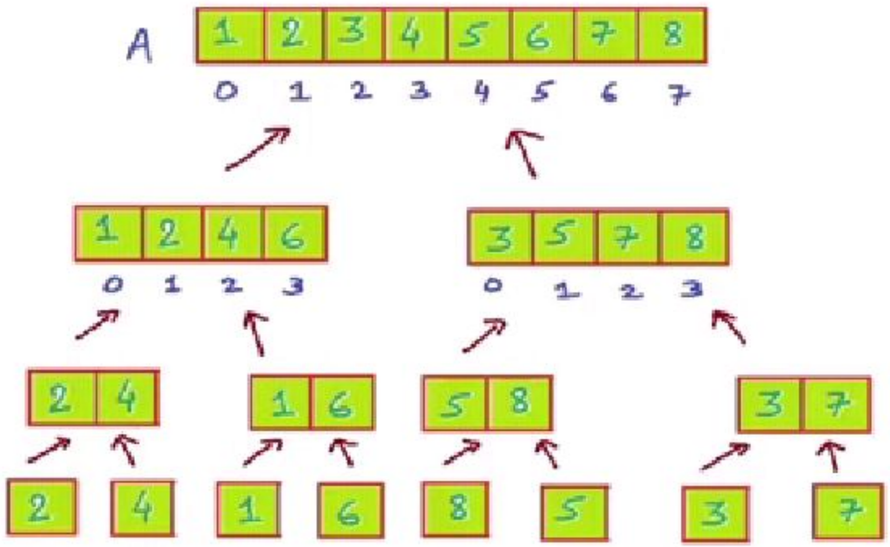
Merge Sort..

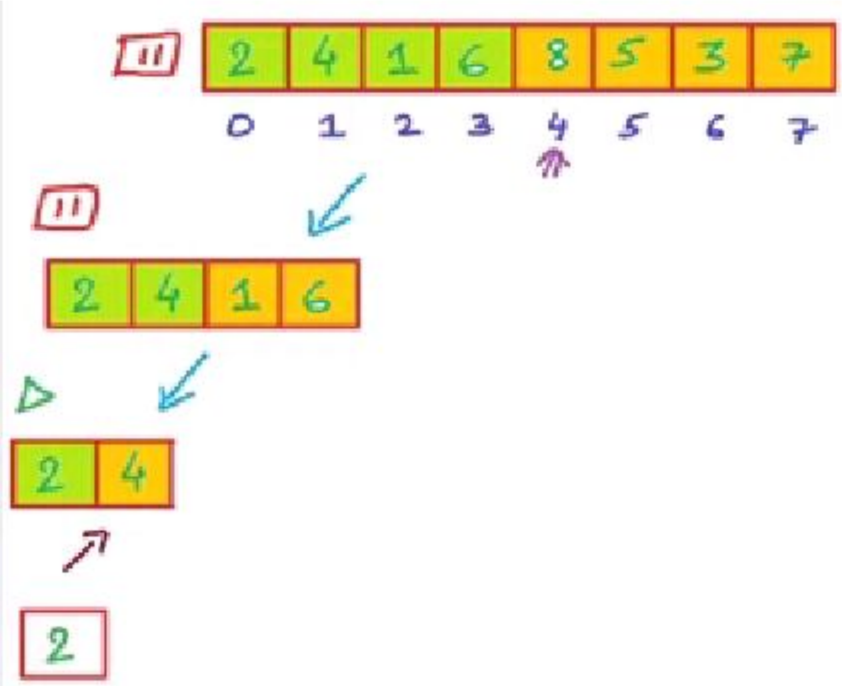

A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
L
2 4 1 6
0 1 2 3
R
8 5 3 7
0 1 2 3


A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
2 4 1 6
0 1 2 3
8 5 3 7
0 1 2 3
2 4
1 6
8 5
3 7


A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
2 4 1 6
0 1 2 3
8 5 3 7
0 1 2 3
2 4
1 6
8 5
3 7
2
4
1
6
8
5
3
7

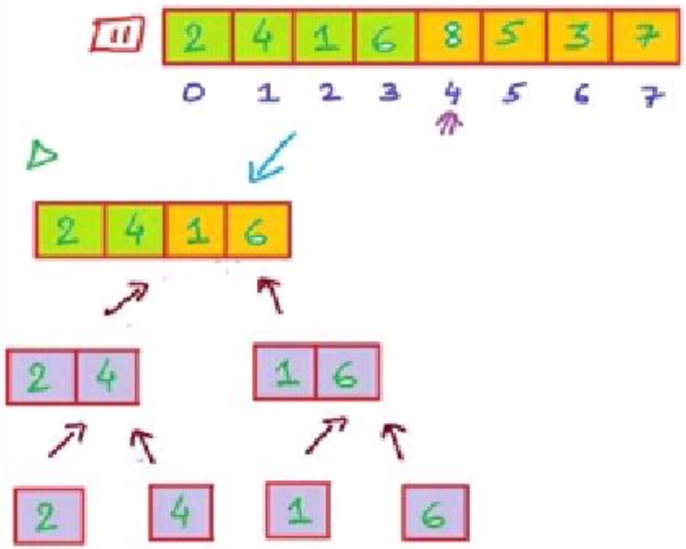

A
2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
2 4 1 6
8 5 3 7
1 2 4 6
3 5 7 8
2 4
1 6
8 5
5 8
3 7

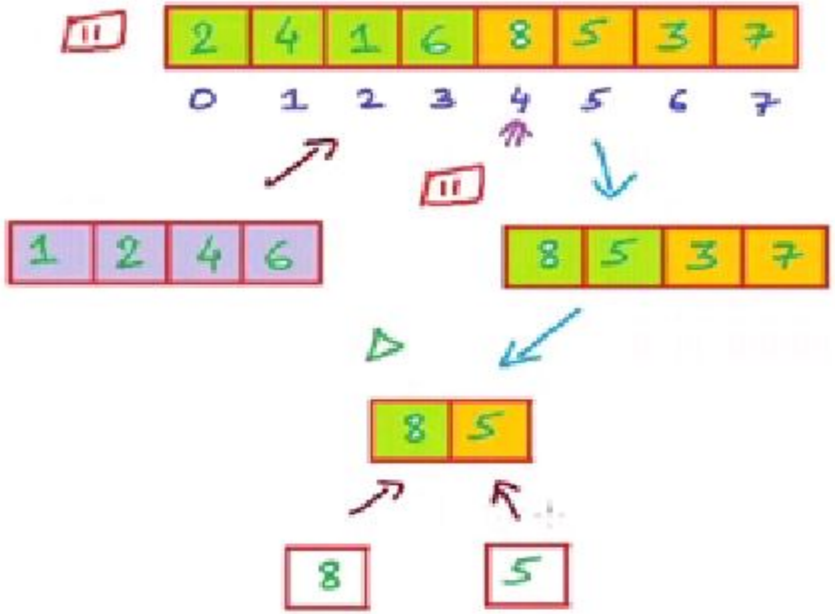

A
1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7
1 2 4 6
0 1 2 3
3 5 7 8
0 1 2 3
2 4
1 6
5 8
3 7
2
4
1
6
8
5
3
7

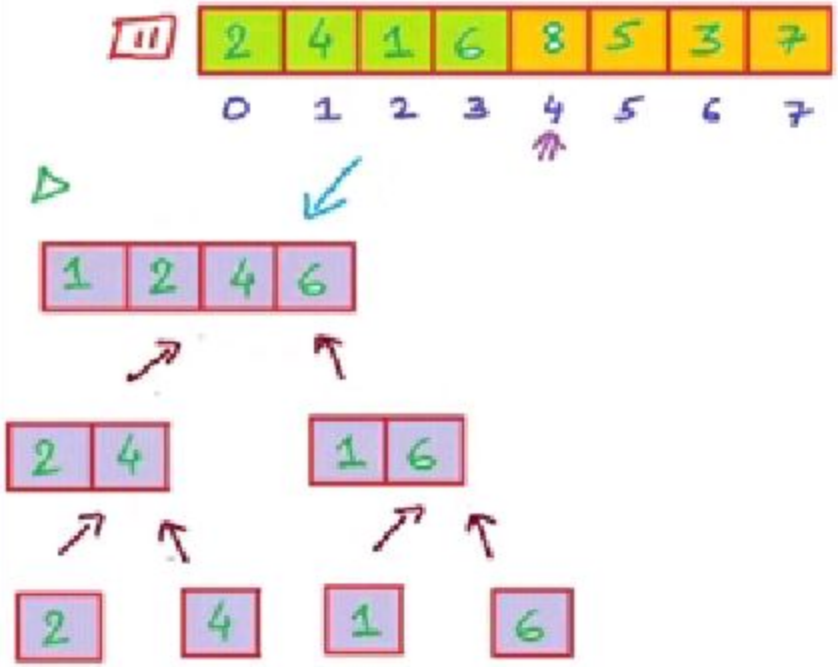


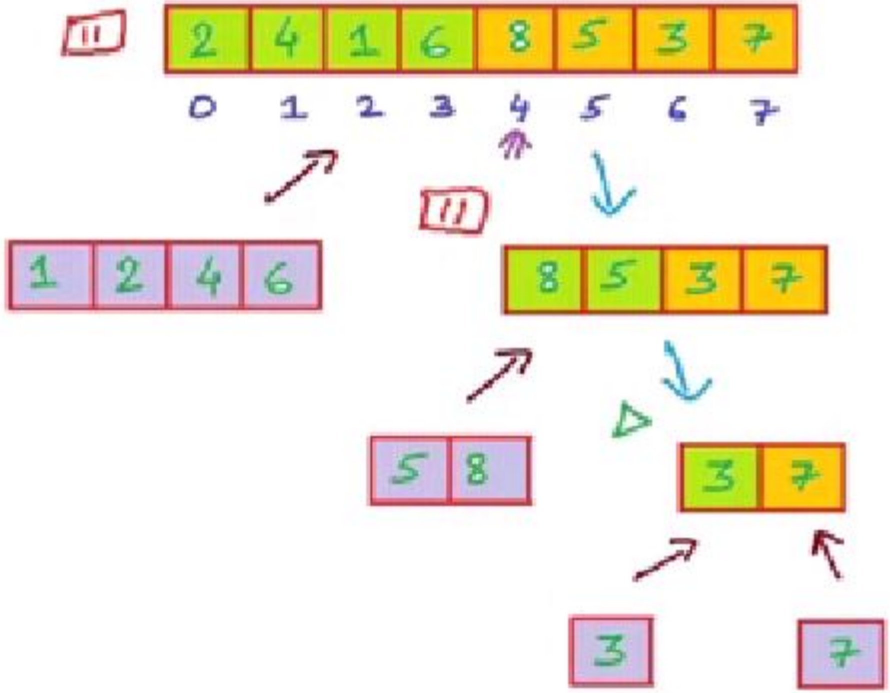
Base Condition

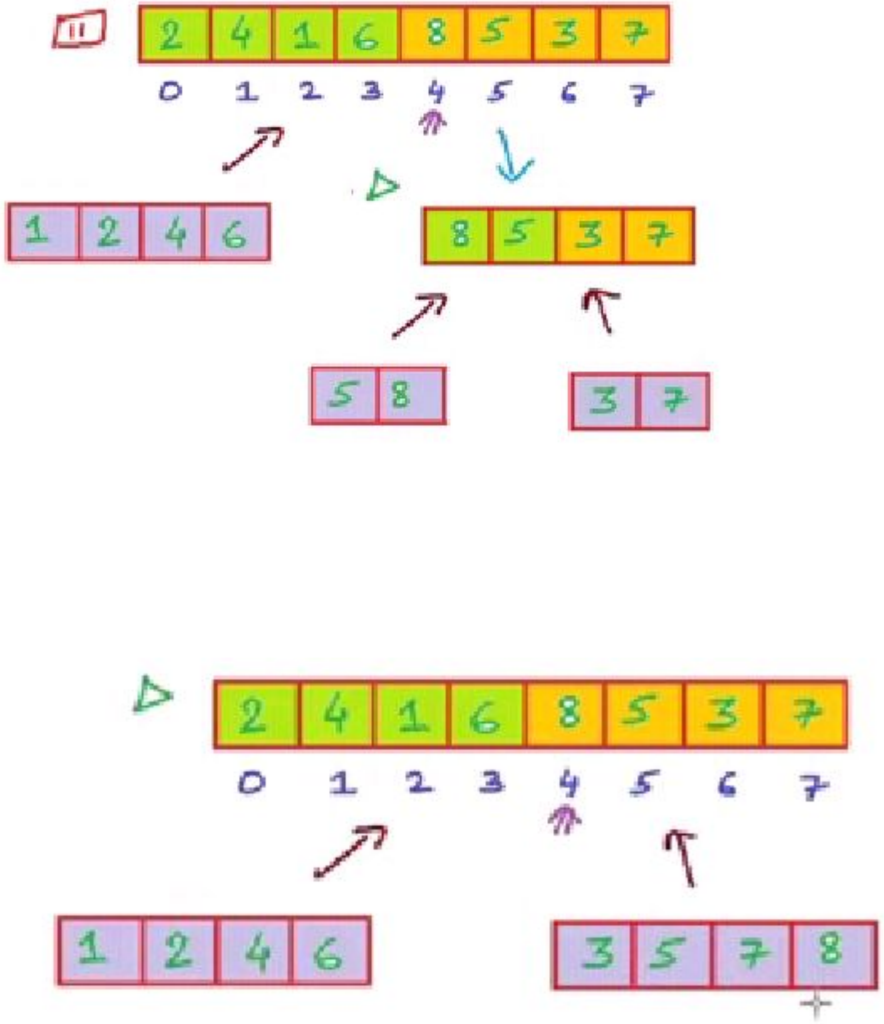
```
Mergesort (A)
{ n ← length(A)
  if (n < 2) return
  mid ← n/2
  left ← array of size(mid)
  right ← array of size(n-mid)
  for i ← 0 to mid-1
     left[i] ← A[i]
  for i ← mid to n-1
     right[i-mid] ← A[i]
  MergeSort(left)
  Mergesort(right)
} Merge(left, right, A)
```


2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
2 4 1 6
2 4
1 6
2
4
1
6


2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
8 5 3 7
1 2 4 6
8 5
8
5


2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
1 2 4 6
2 4
1 6
2
4
1
6


2 4 1 6 8 5 3 7
0 1 2 3 4 5 6 7
1 2 4 6
8 5 3 7
5 8
3 7
3
7




```
Mergesort(A)
{ n ← length(A)
   if (n < 2) return
   mid ← n/2
   left ← array of size(mid)
   right ← array of size(n-mid)
   for i ← 0 to mid-1
      left[i] ← A[i]
   for i ← mid to n-1
      right[i-mid] ← A[i]
   Mergesort(left)
   Mergesort(right)
}  Merge(left, right, A)
```

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

```
Mergesort (A)
{ n ← length(A)
  if (n < 2) return                     ← base Condition
  mid ← n/2
  left ← array of size(mid)
  right ← array of size(n-mid)
  for i ← 0 to mid-1
      left[i] ← A[i]
  for i ← mid to n-1
      right[i-mid] ← A[i]
  Mergesort (left)                      ← Recursive calls
  Mergesort (right)
  Merge(left, right, A)                 ← Merge sorted halves
}
```

98	23	45	14	6	67	33	42

98	23	45	14	6	67	33	42

98	23	45	14

6	67	33	42

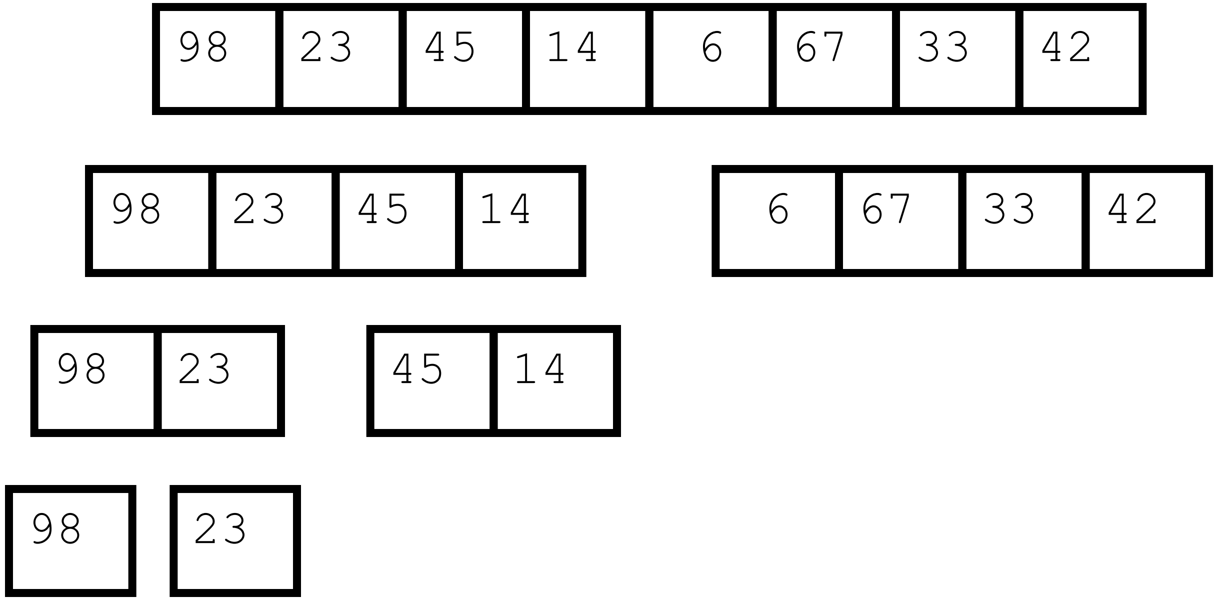
98	23	45	14	6	67	33	42

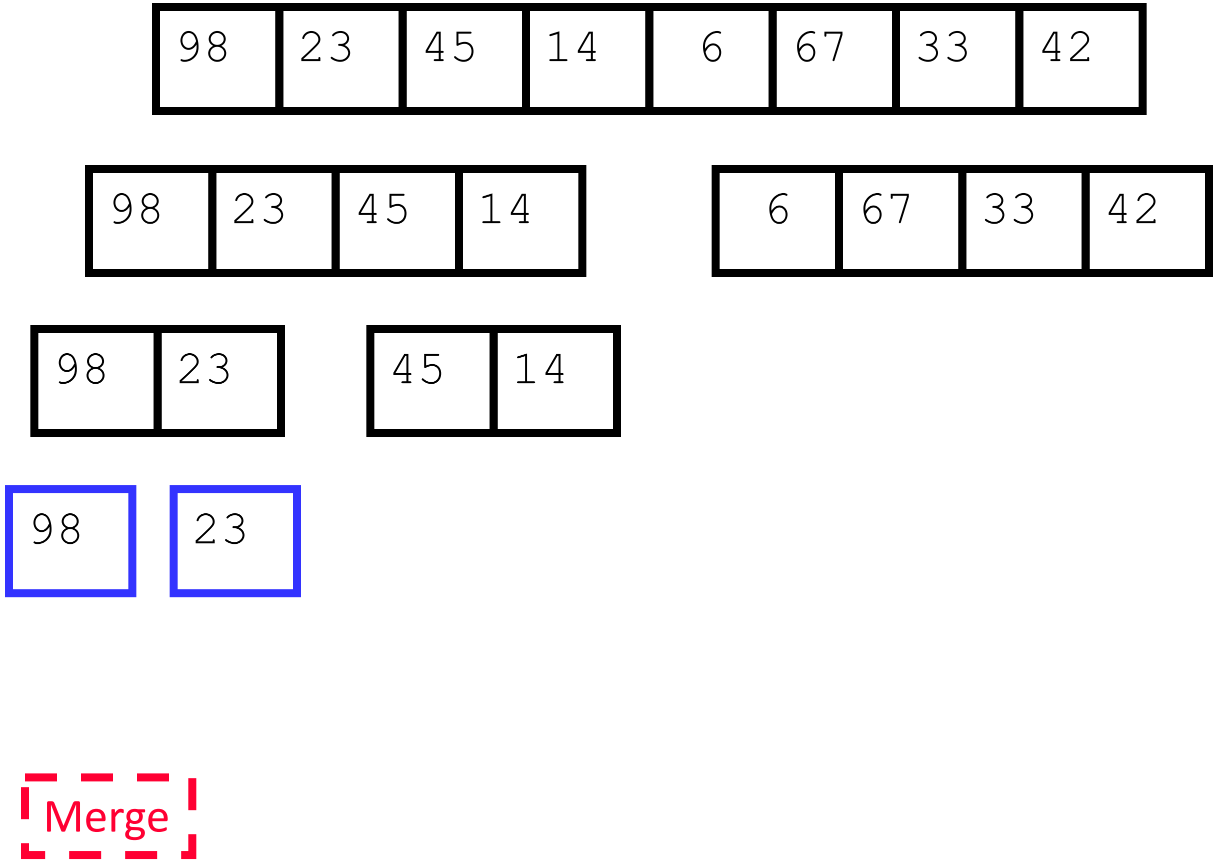
98	23	45	14

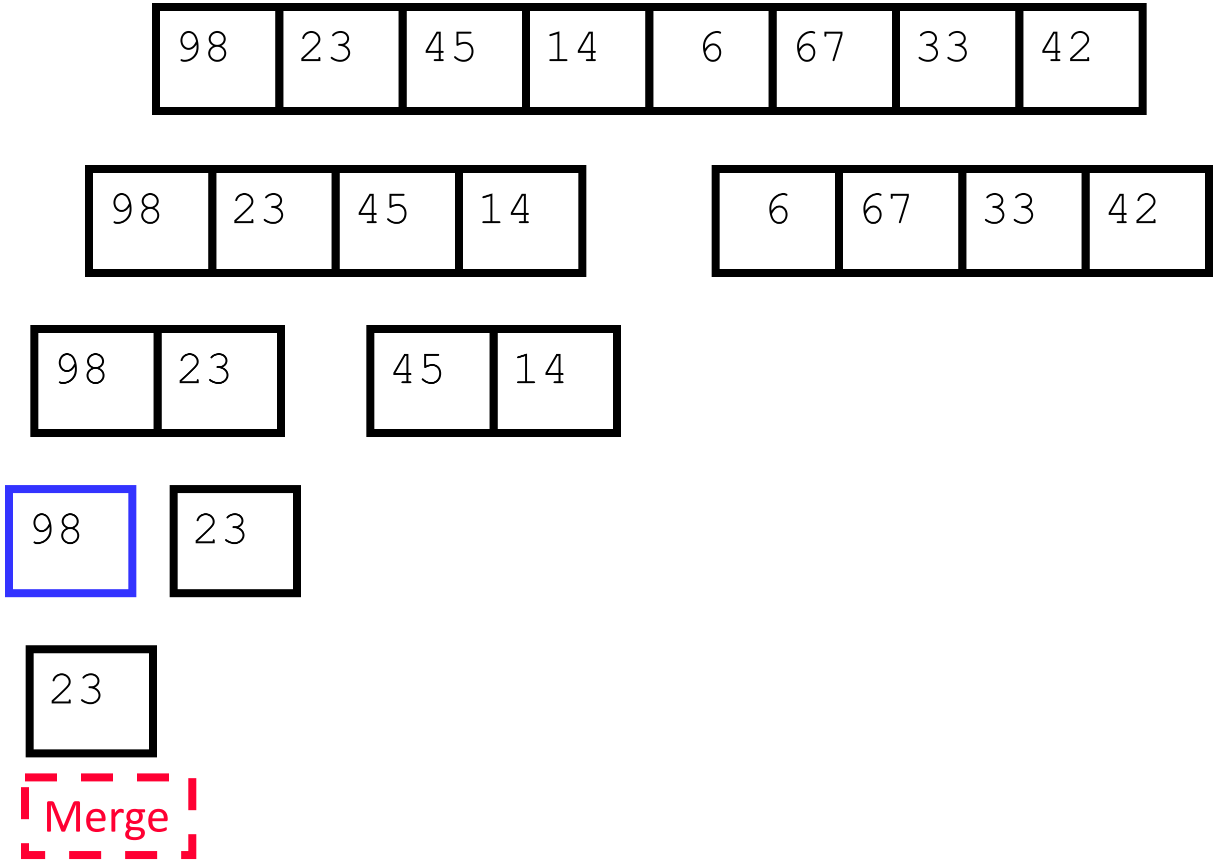
6	67	33	42

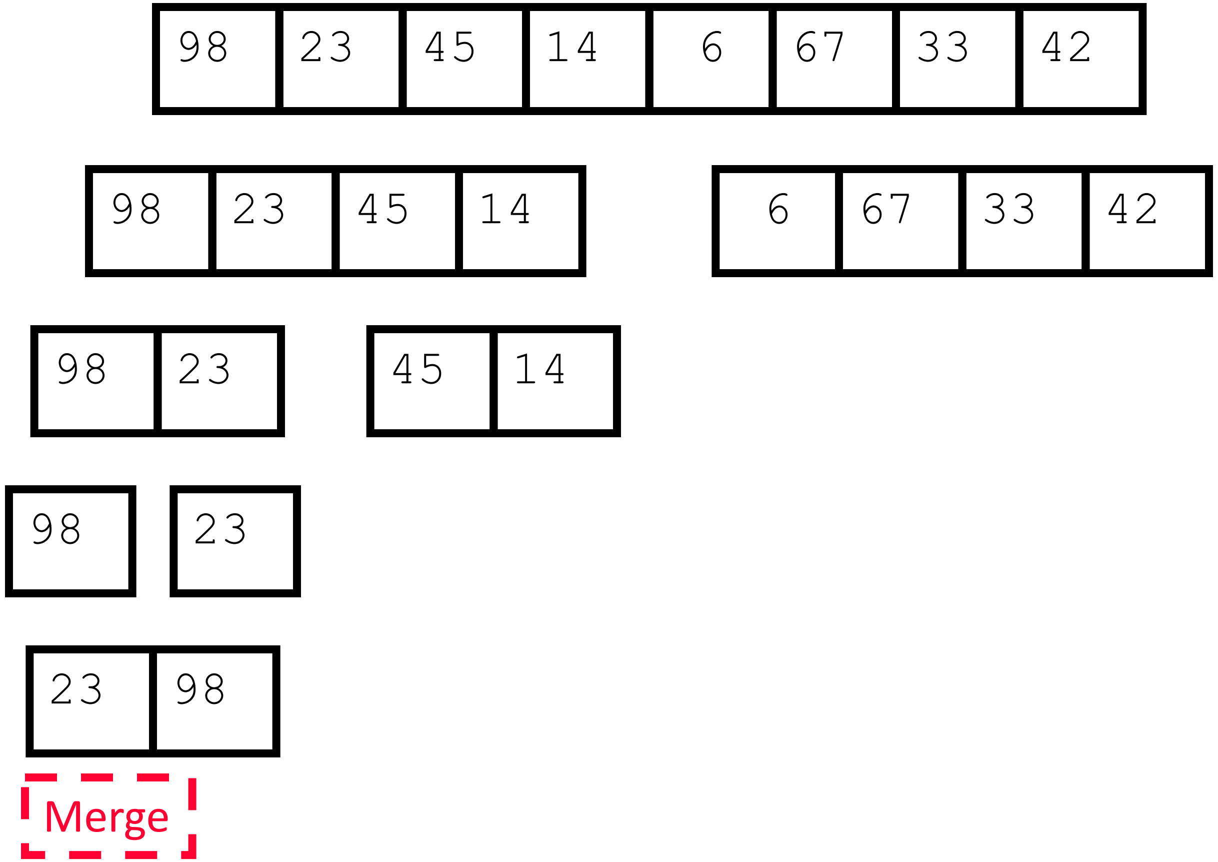
98	23

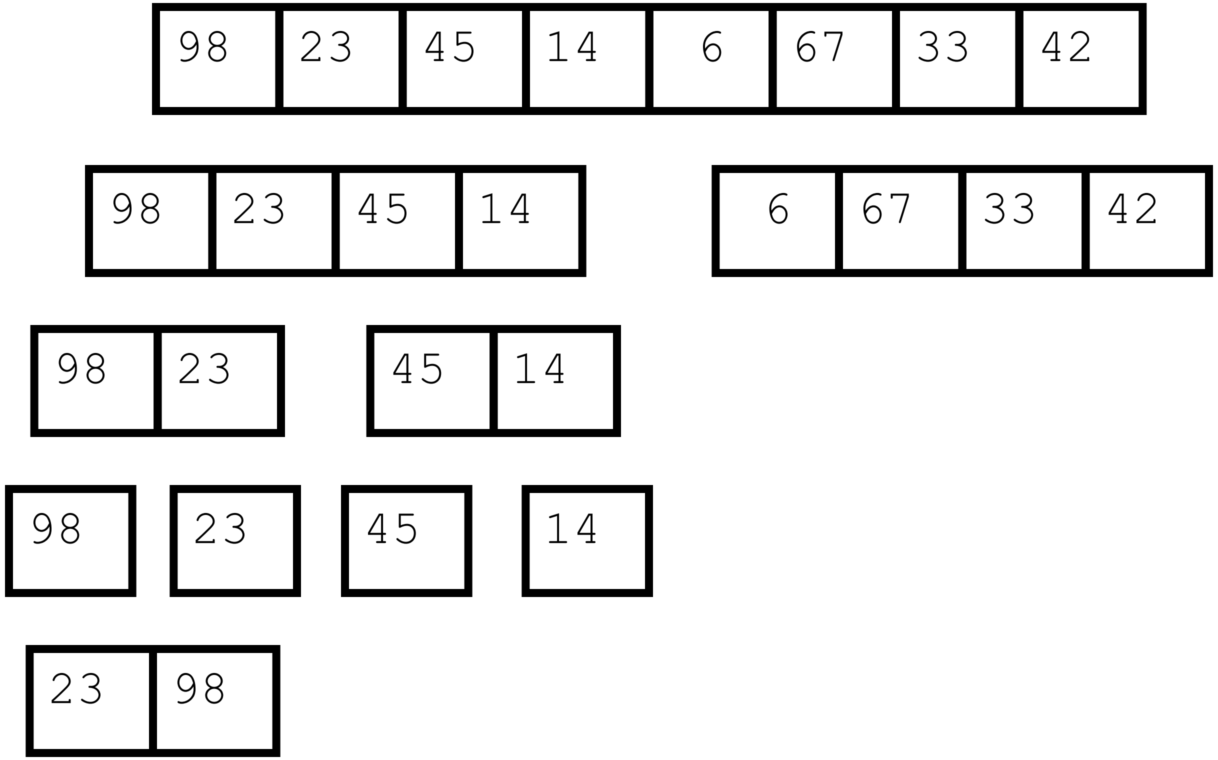
45	14

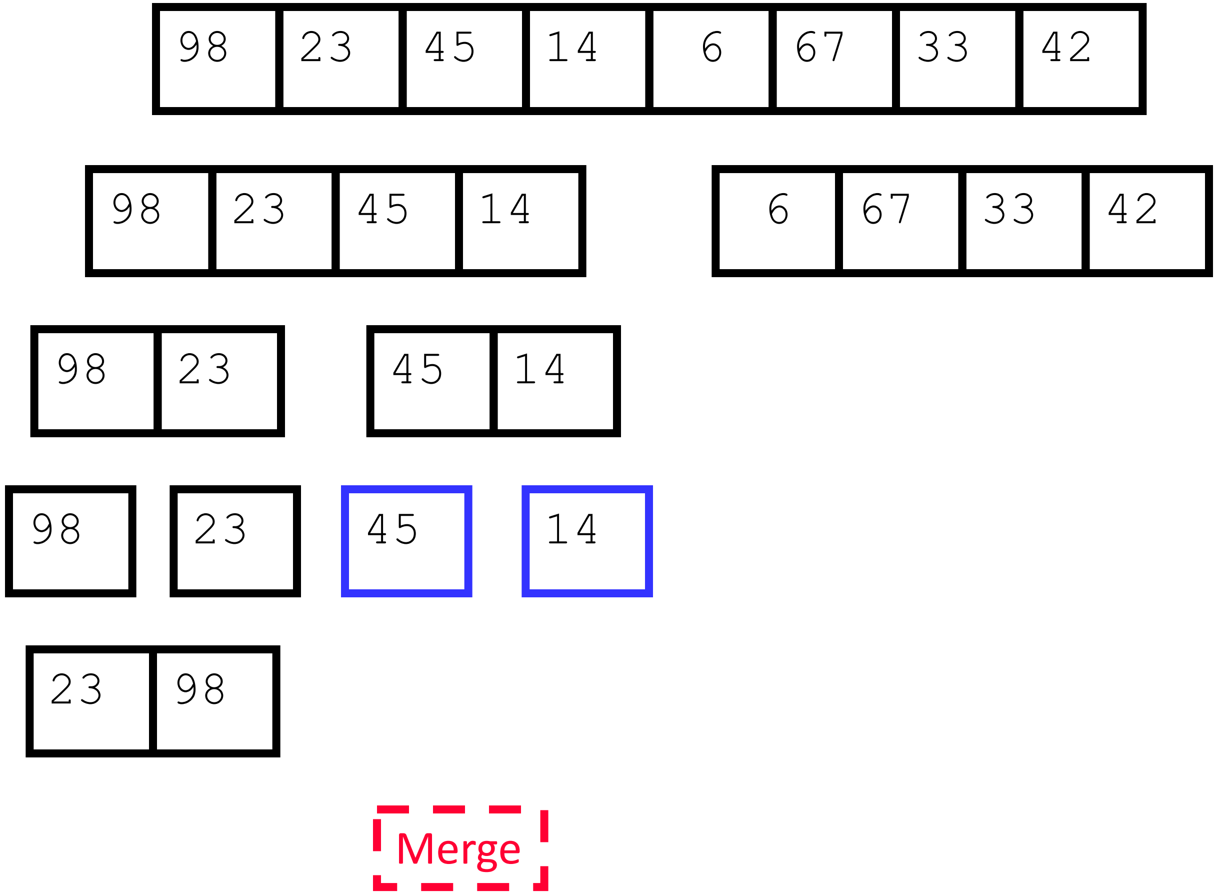

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23

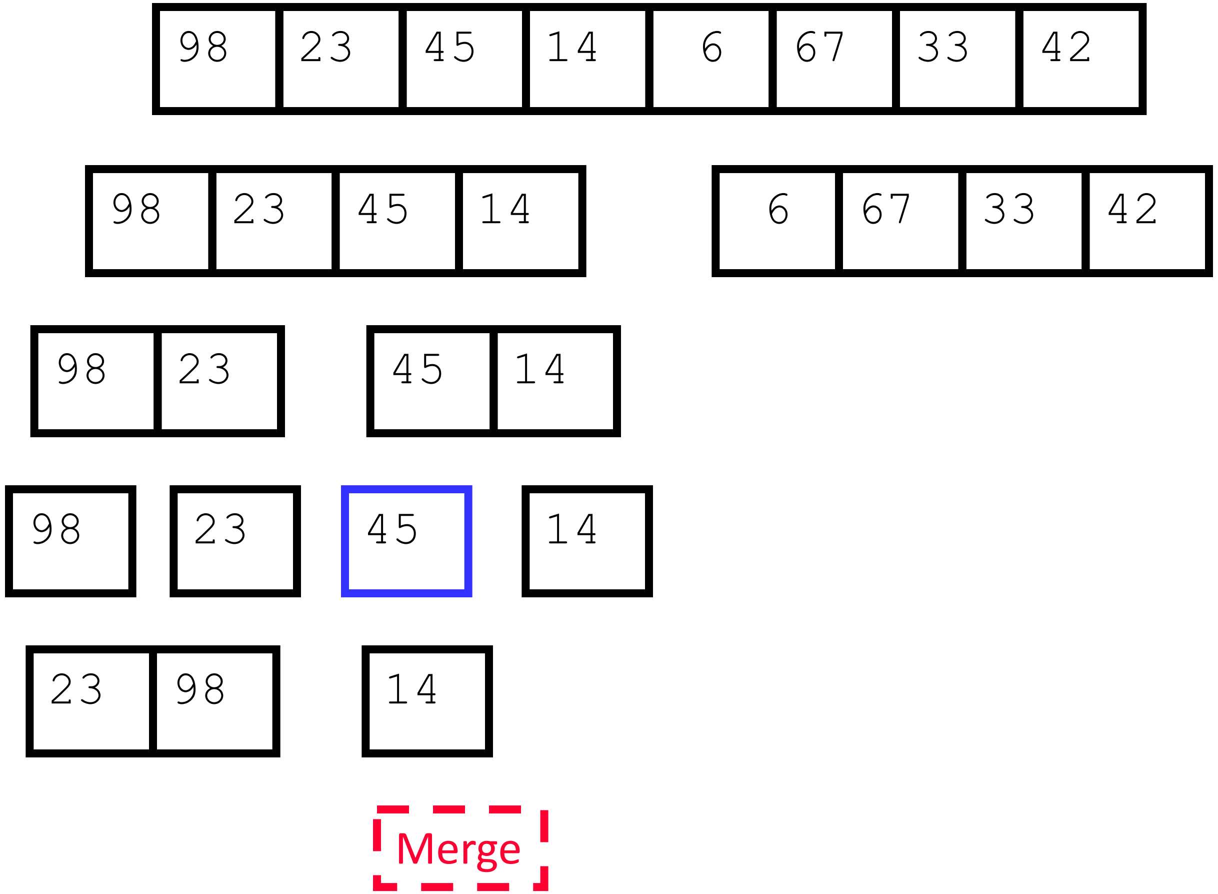

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
Merge

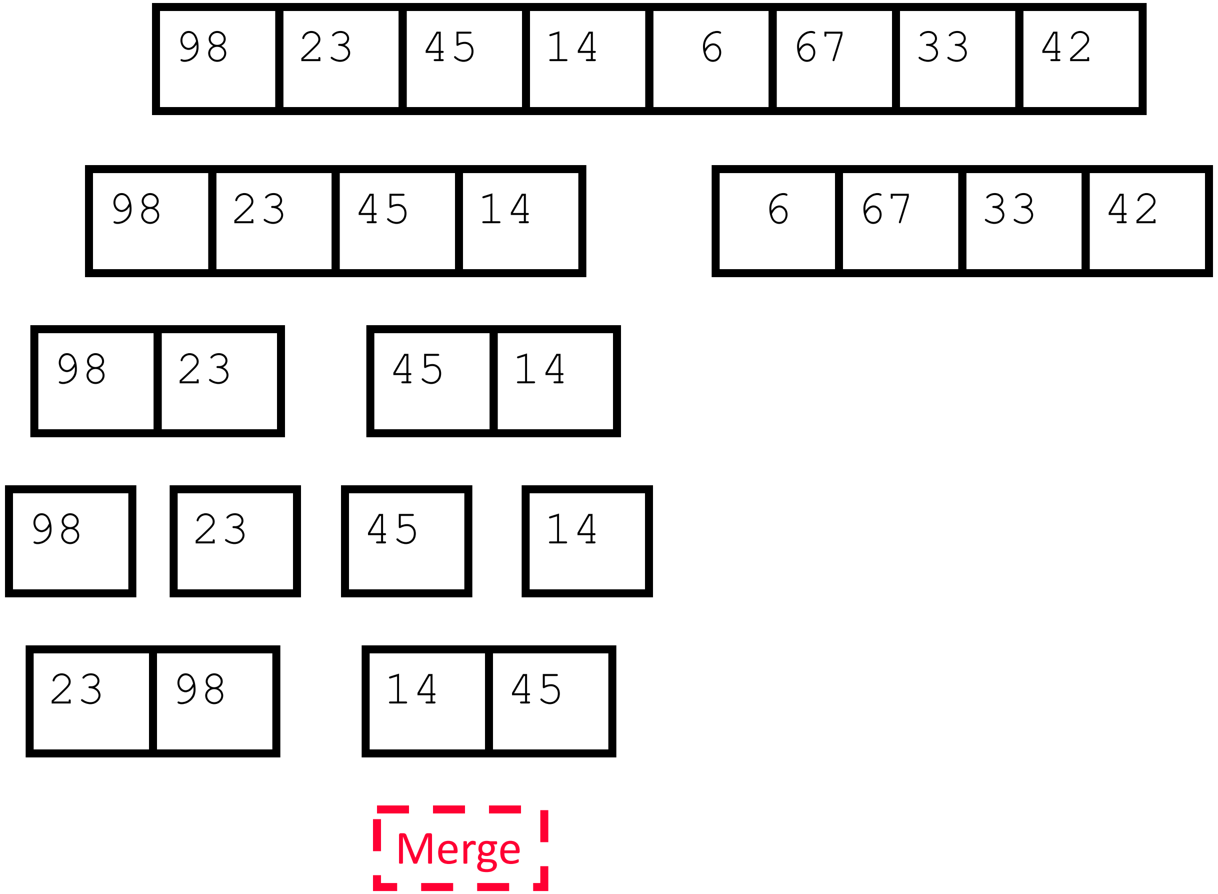

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23
Merge

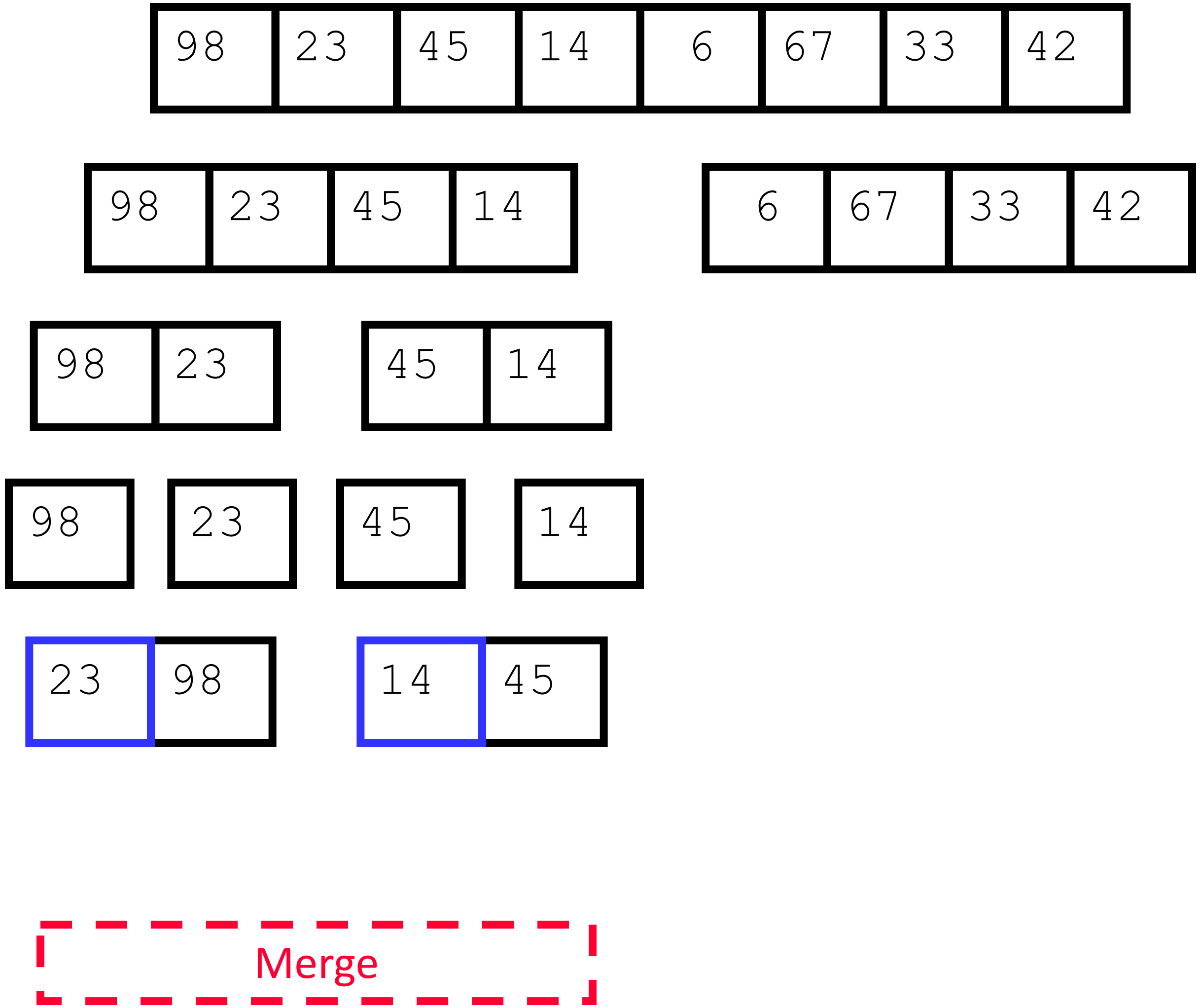

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
23 98
Merge

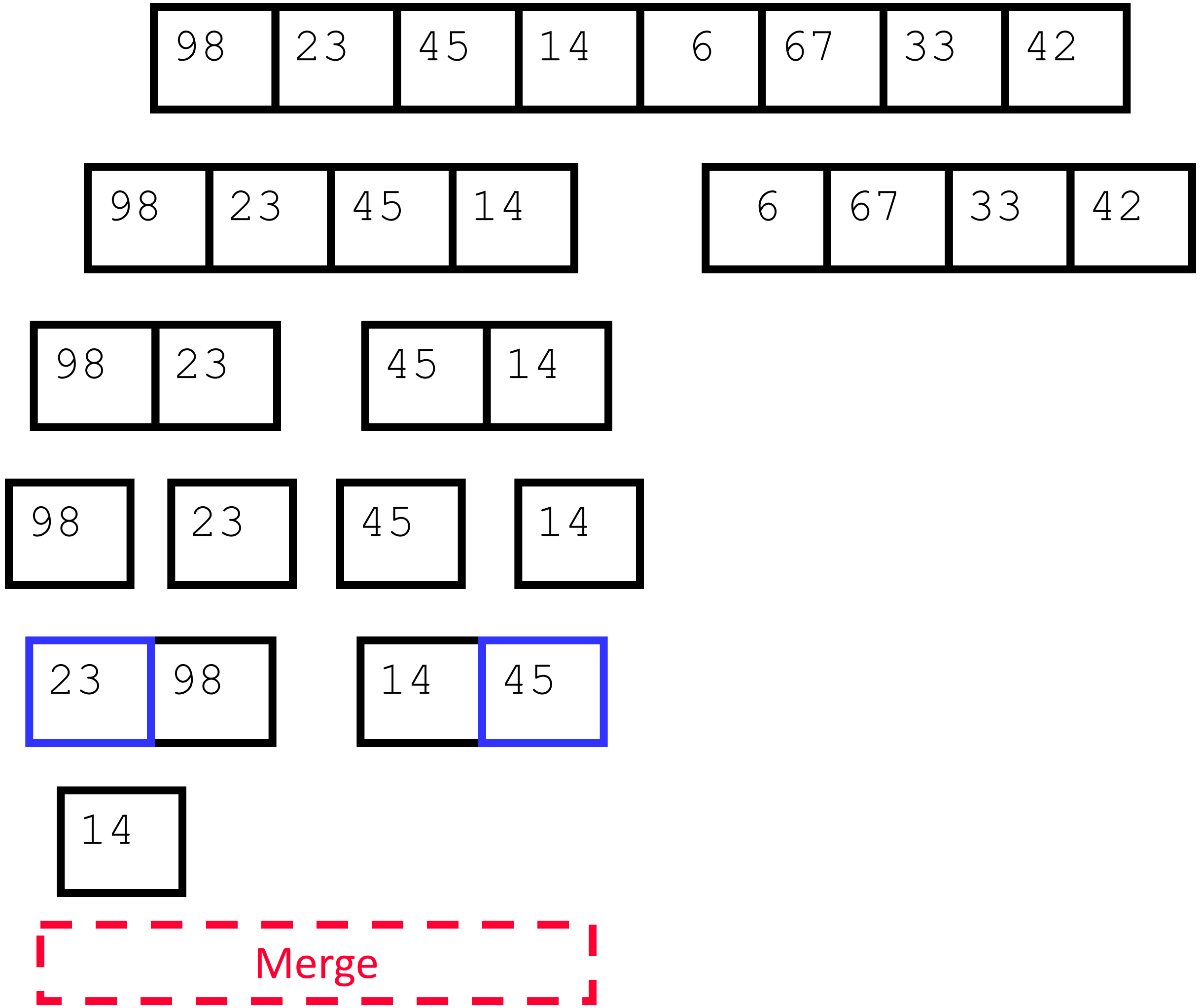

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge

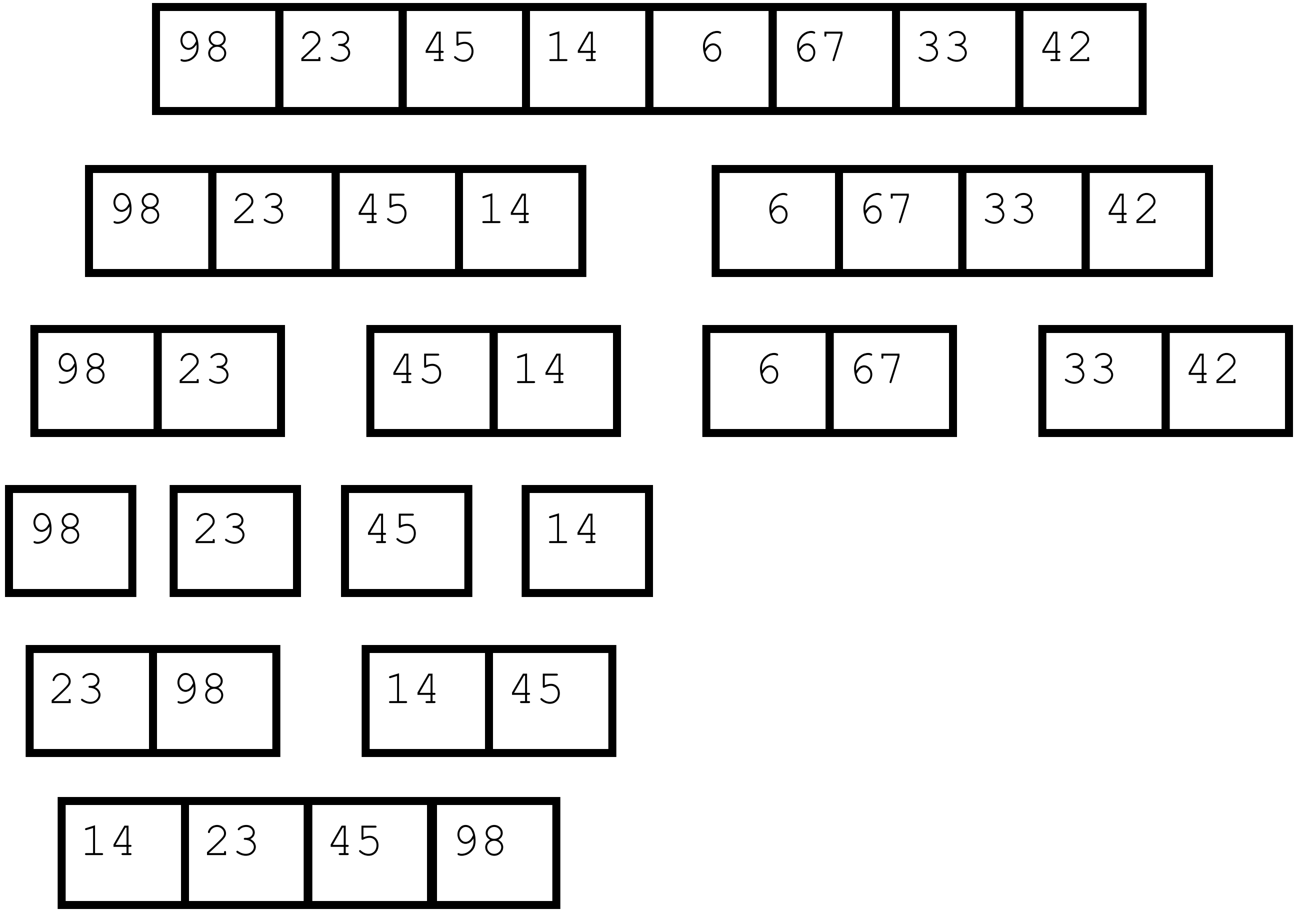

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14
Merge

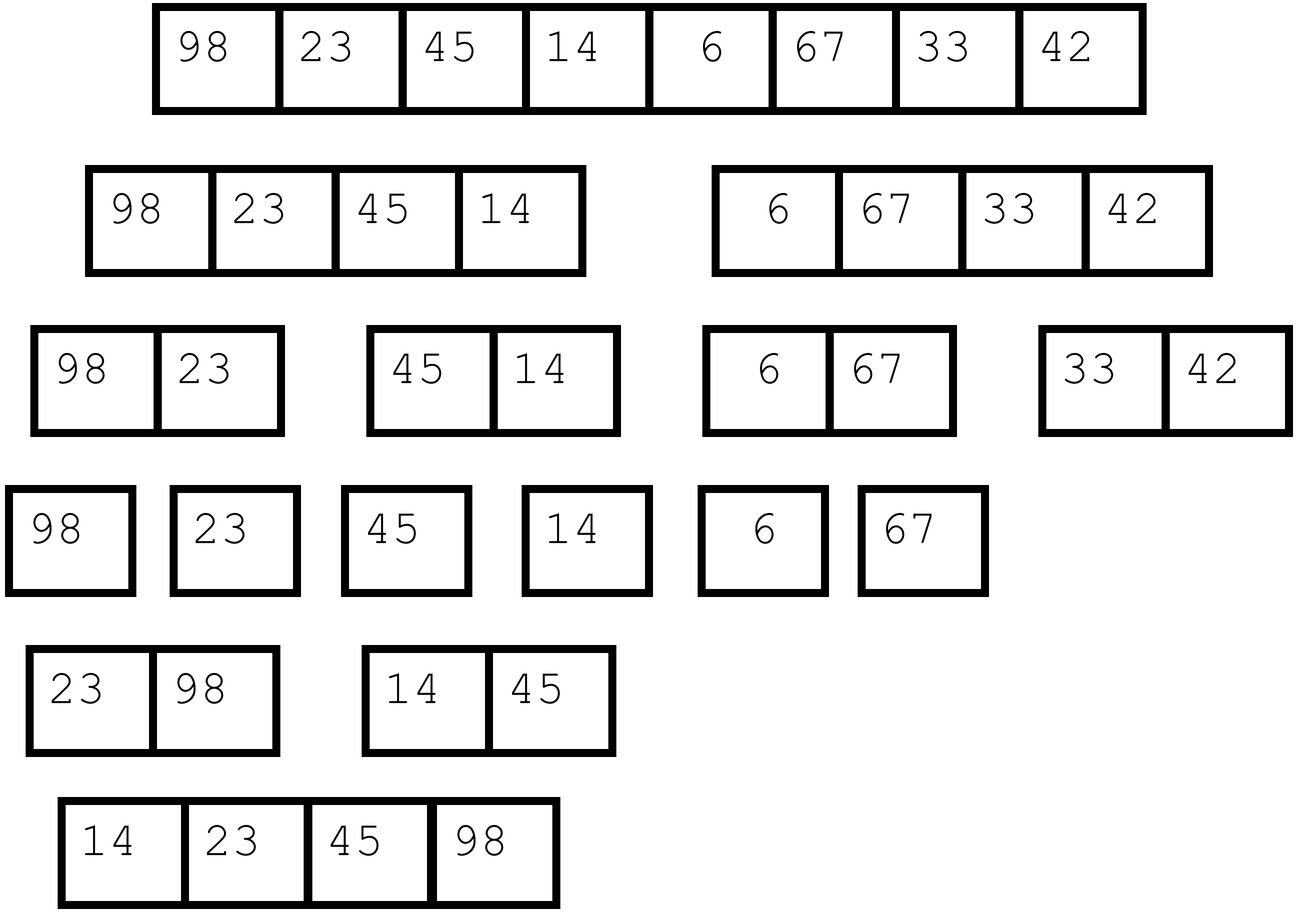

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23
Merge

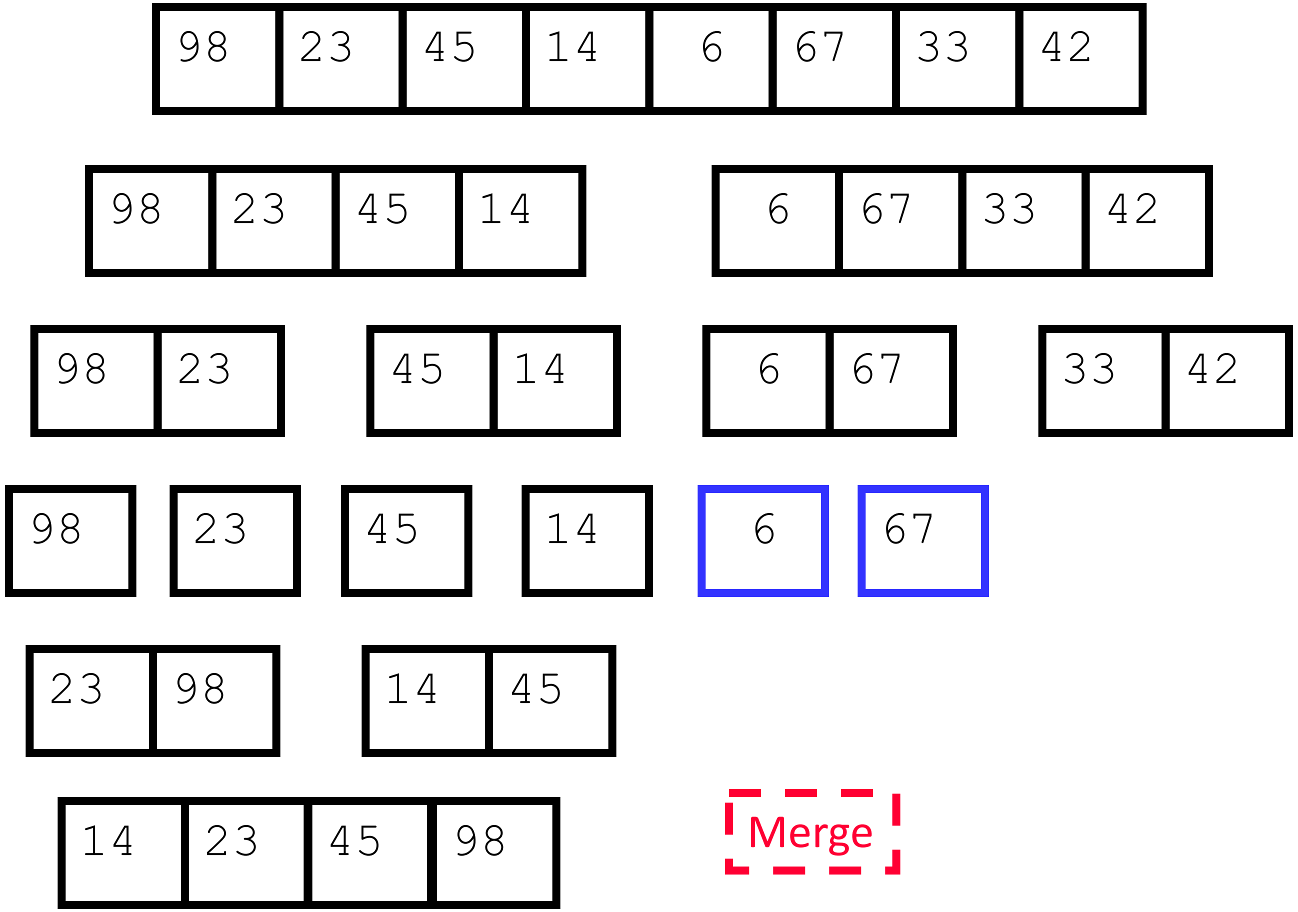

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45
Merge

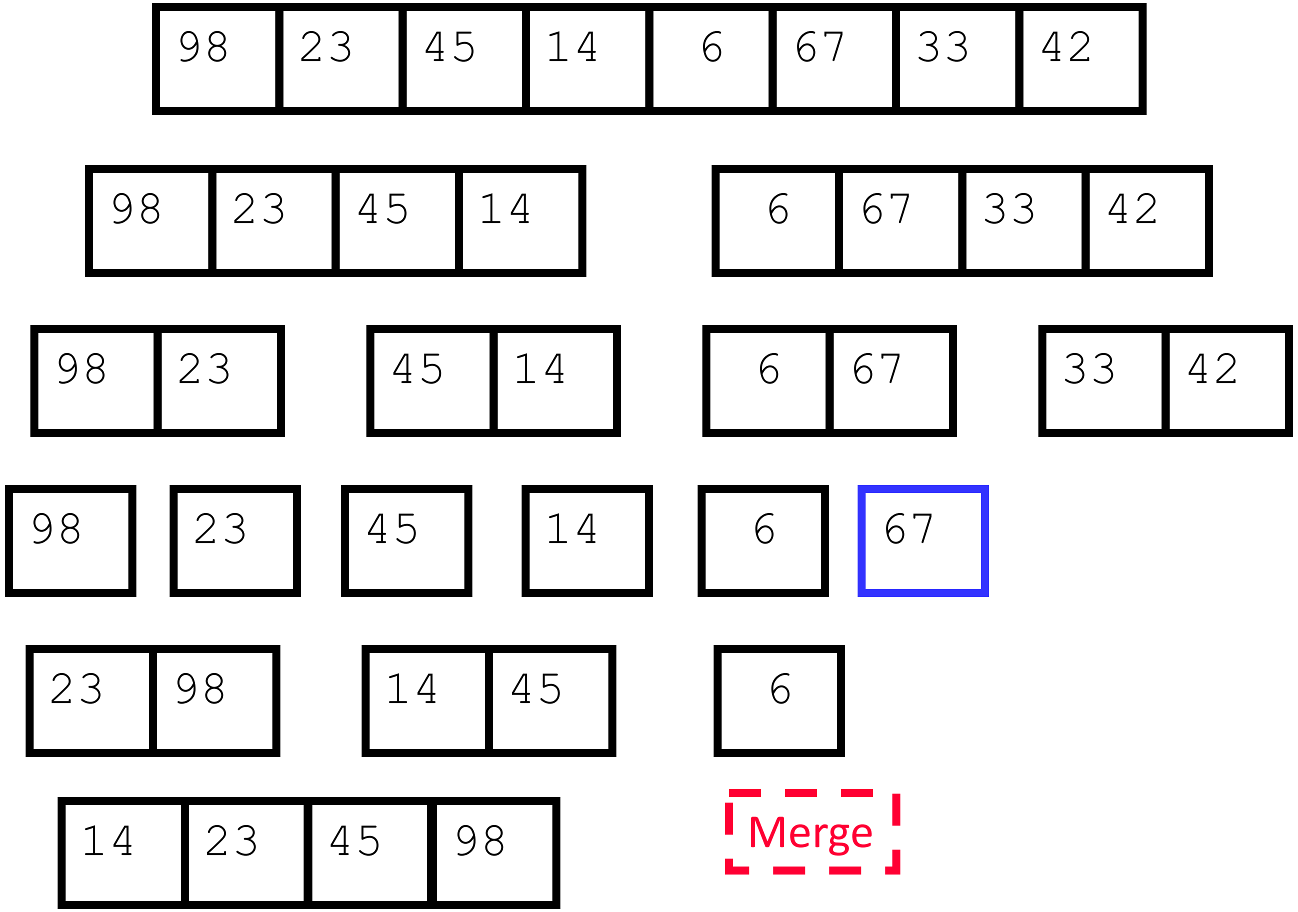

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
98
23
45
14
23 98
14 45
14 23 45 98
Merge

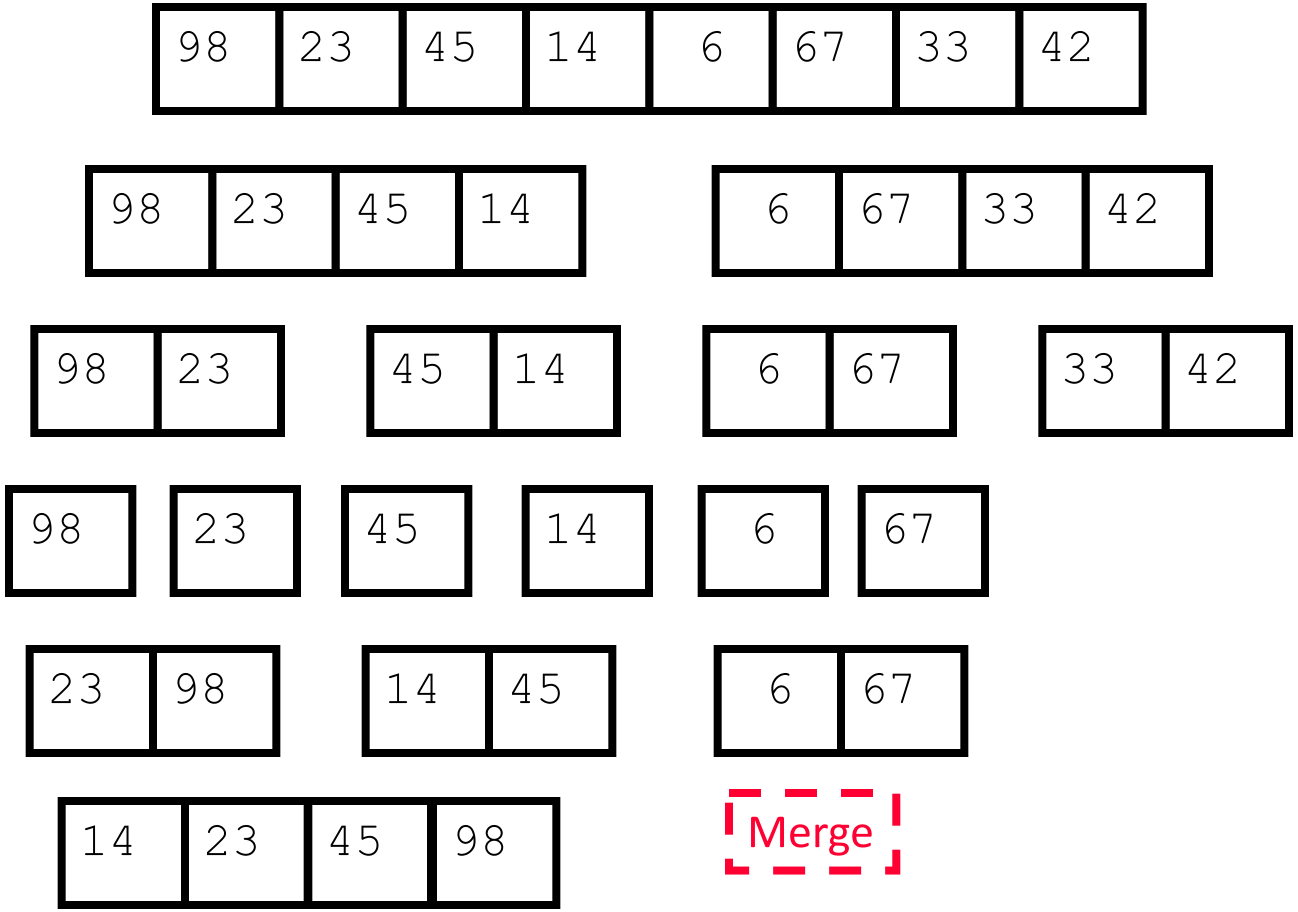


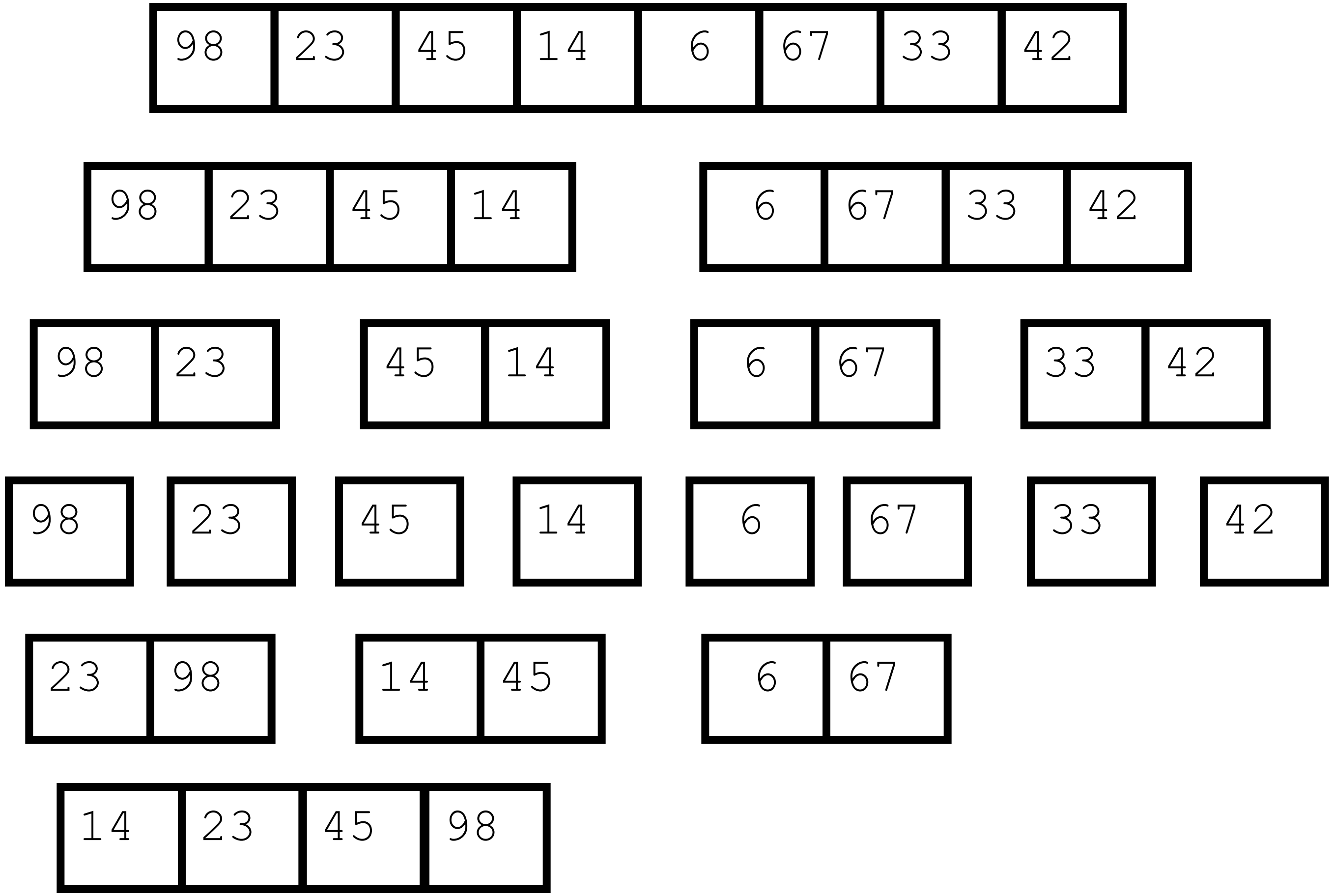
98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
23 98
14 45
14 23 45 98

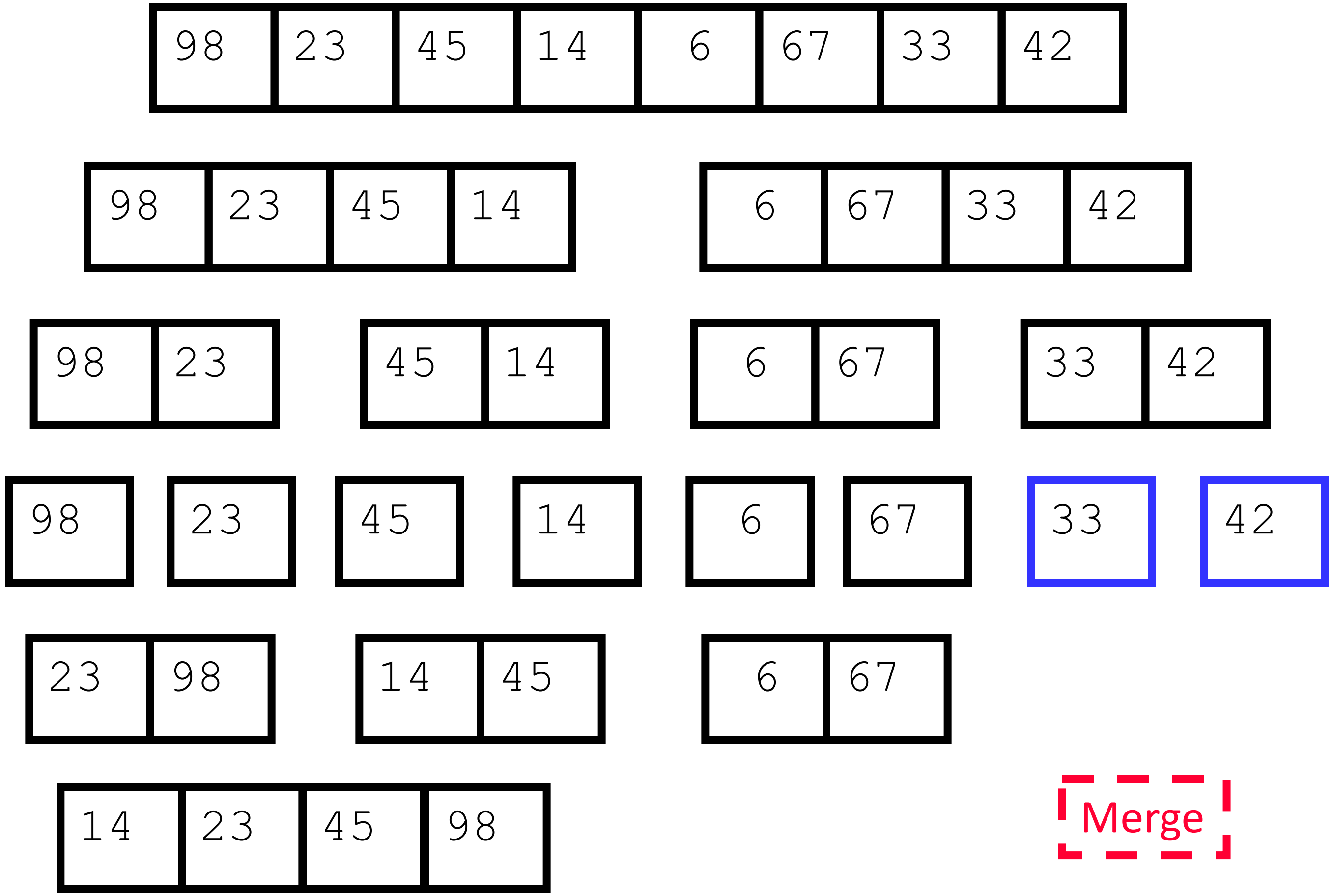

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98

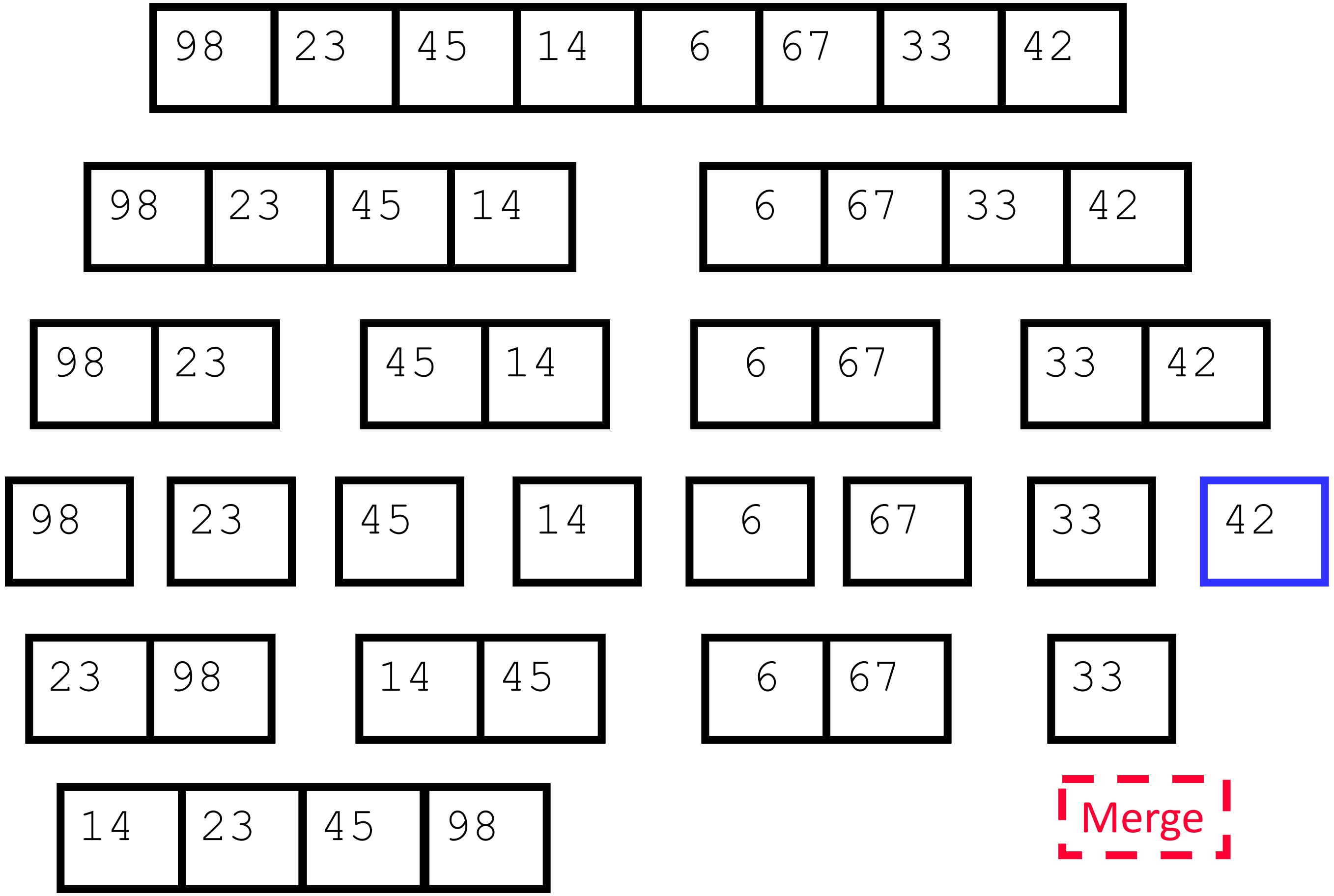

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
14 23 45 98
Merge

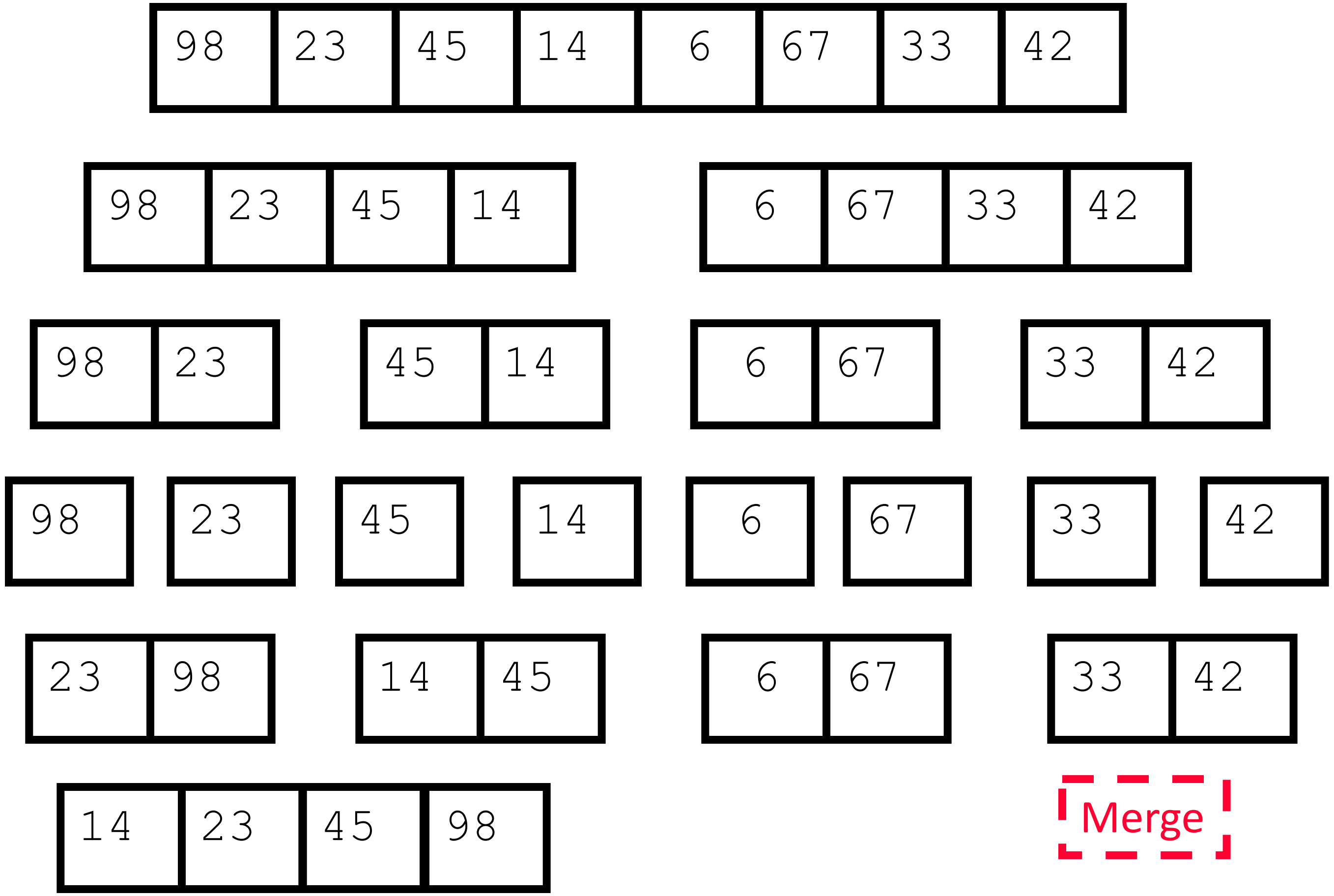

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6
14 23 45 98
Merge

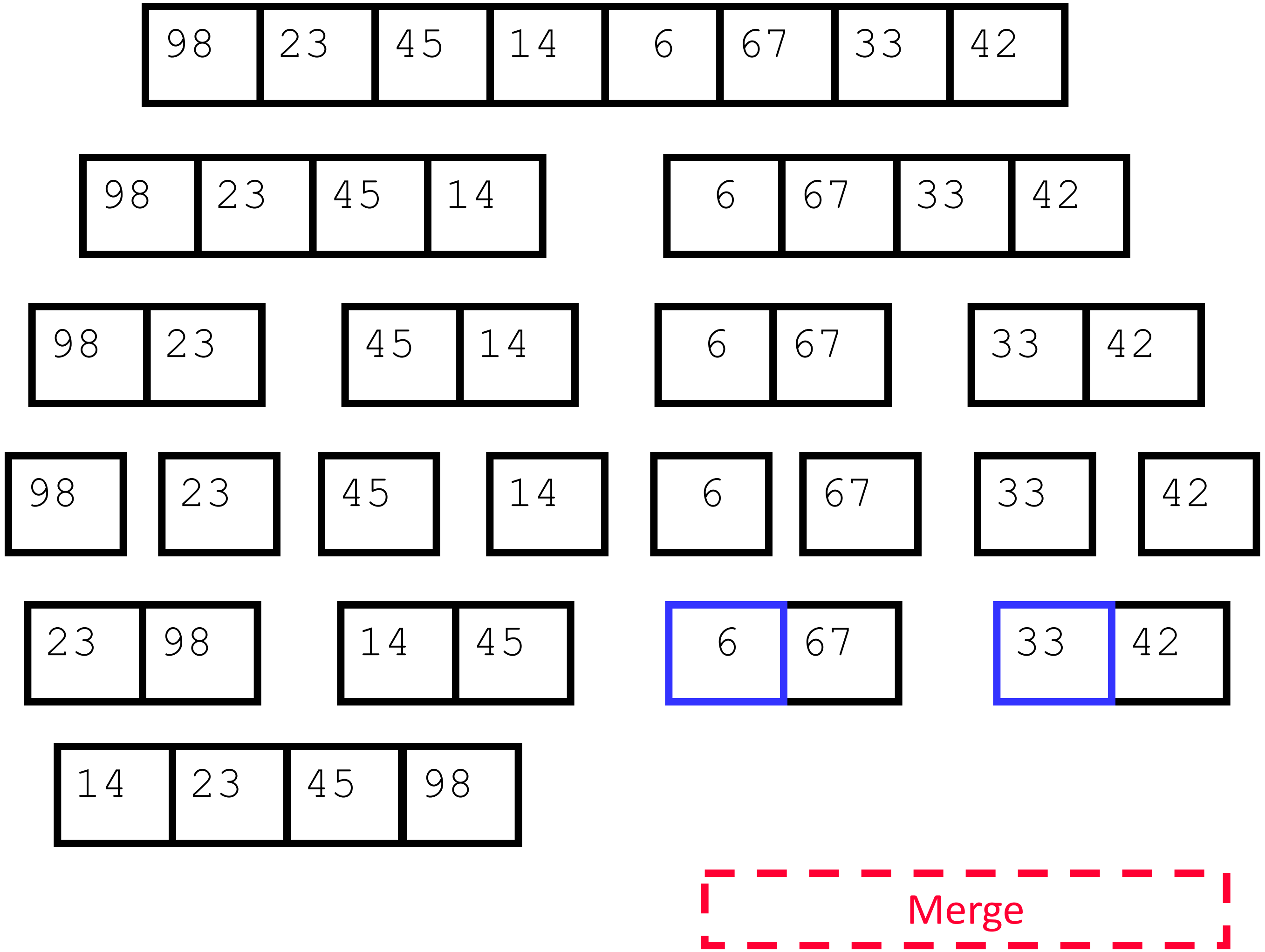

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
23 98
14 45
6 67
14 23 45 98
Merge

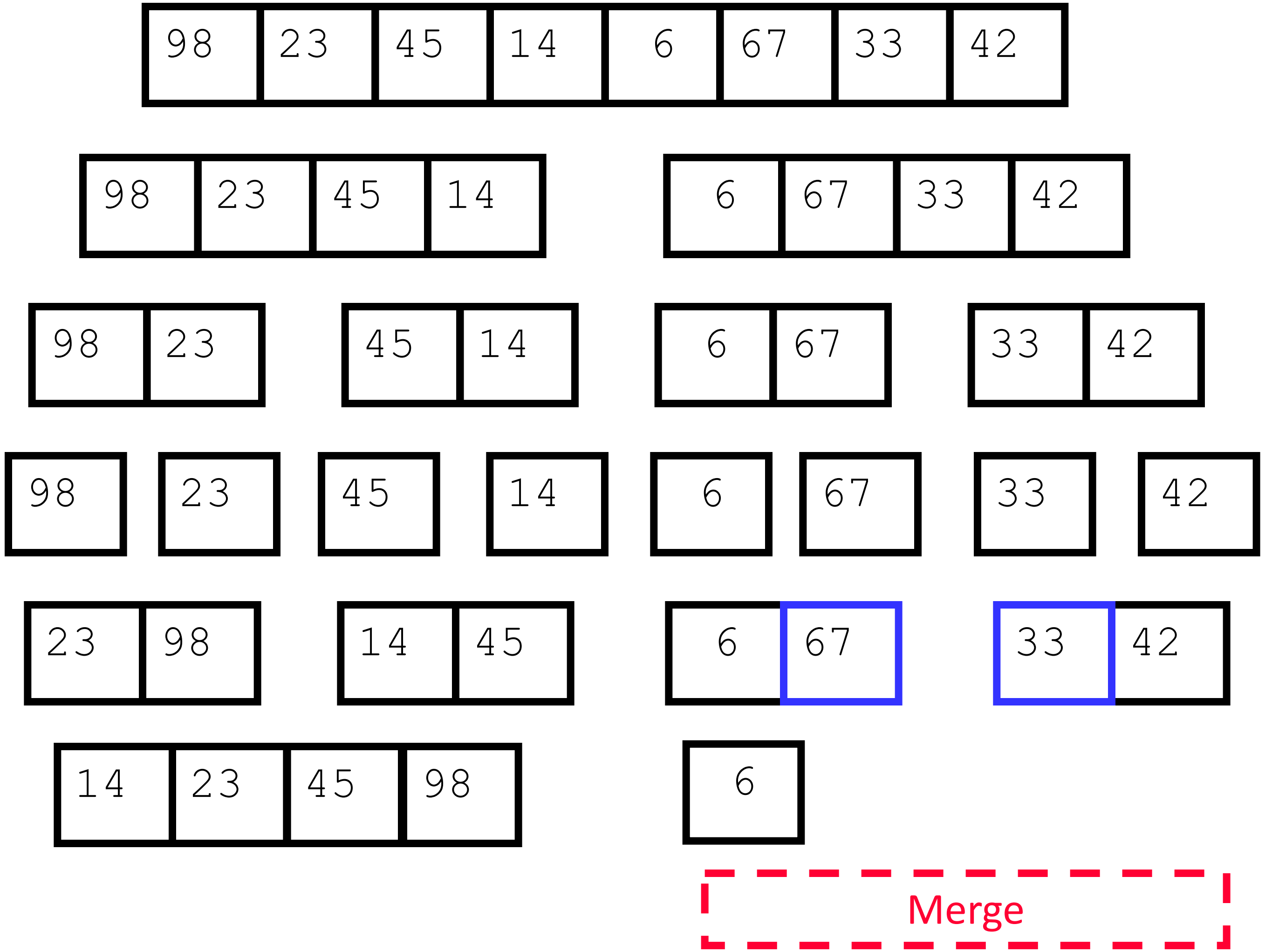

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
14 23 45 98
Merge

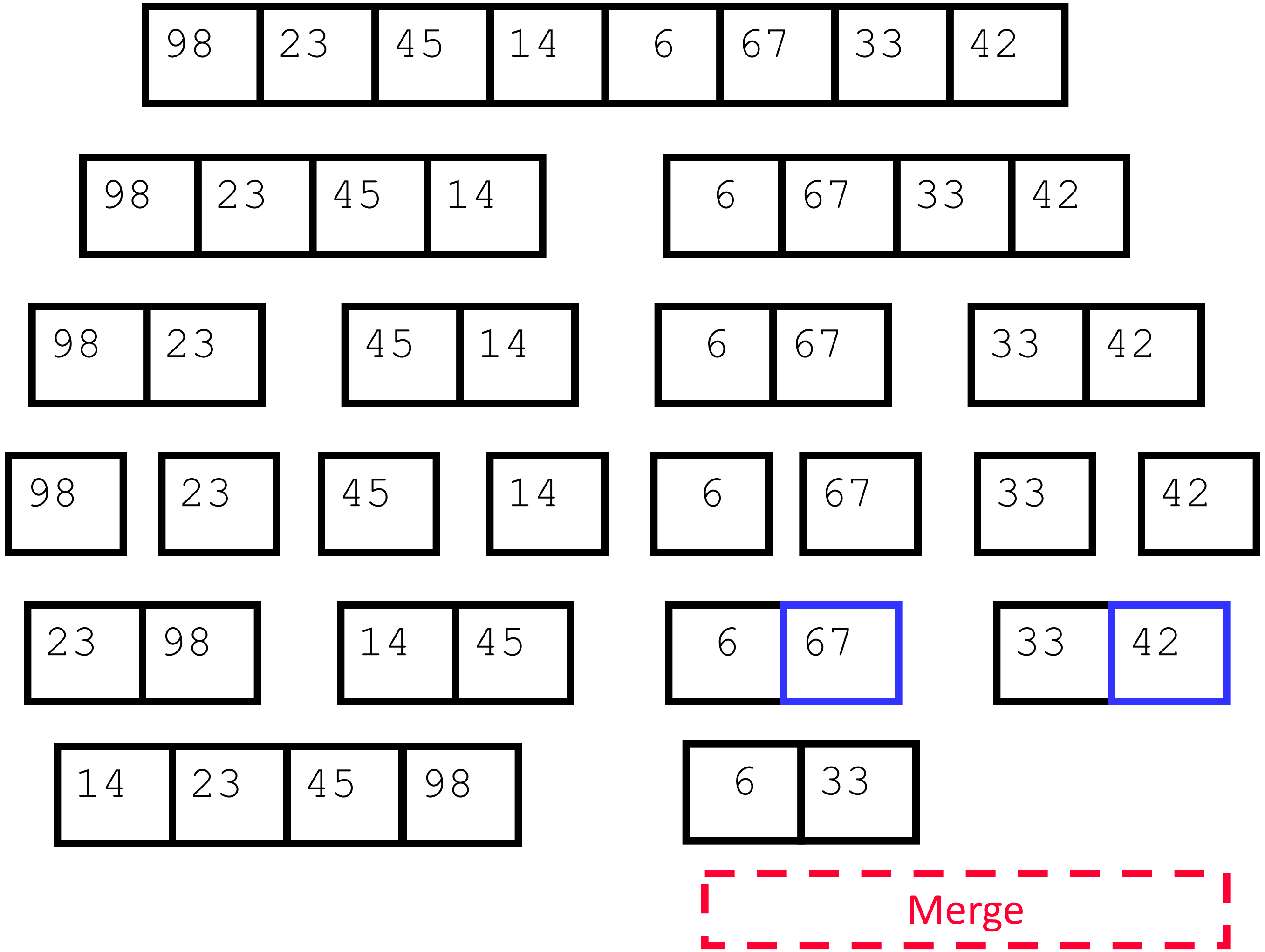

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33
14 23 45 98
Merge

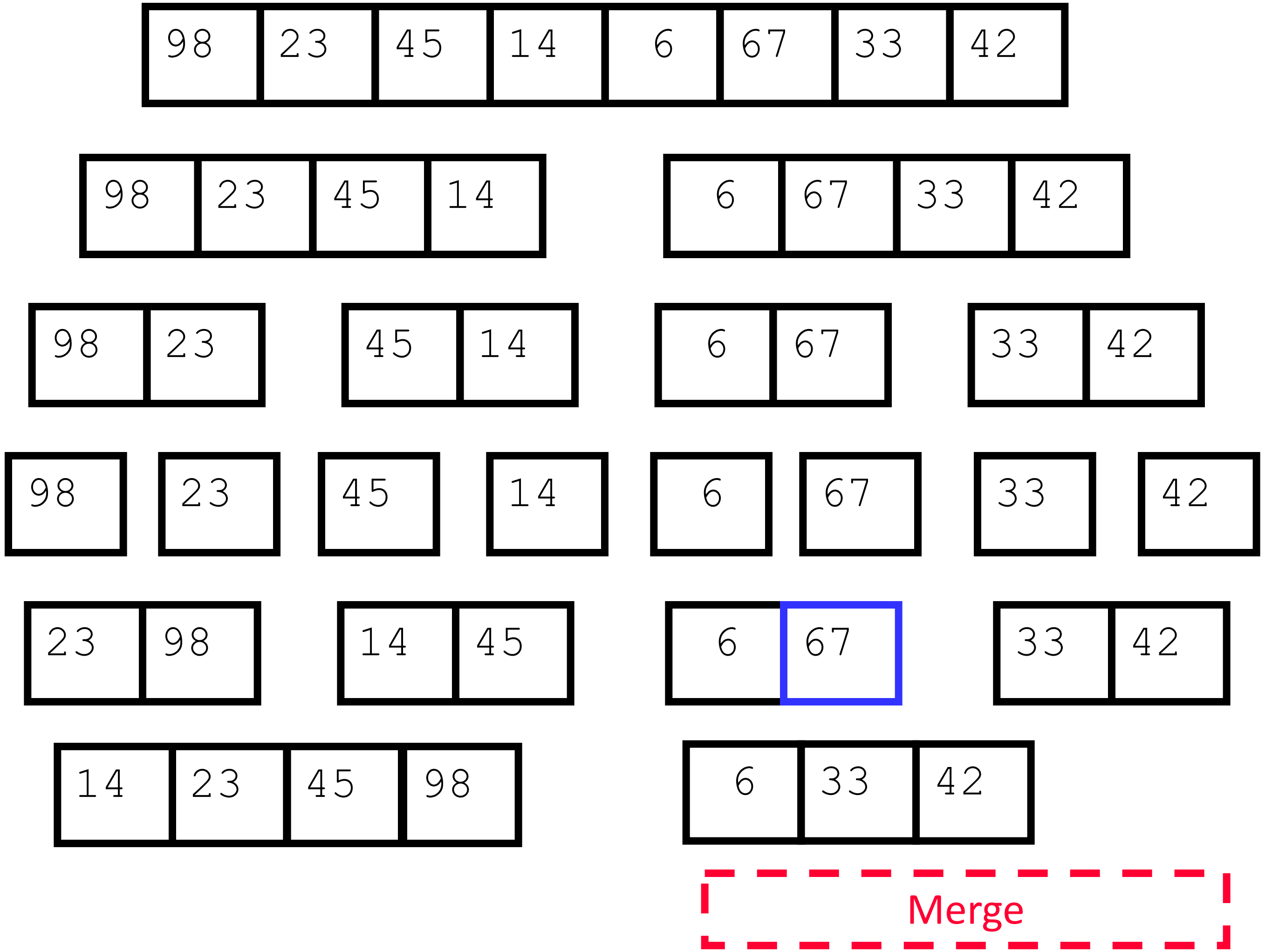

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge

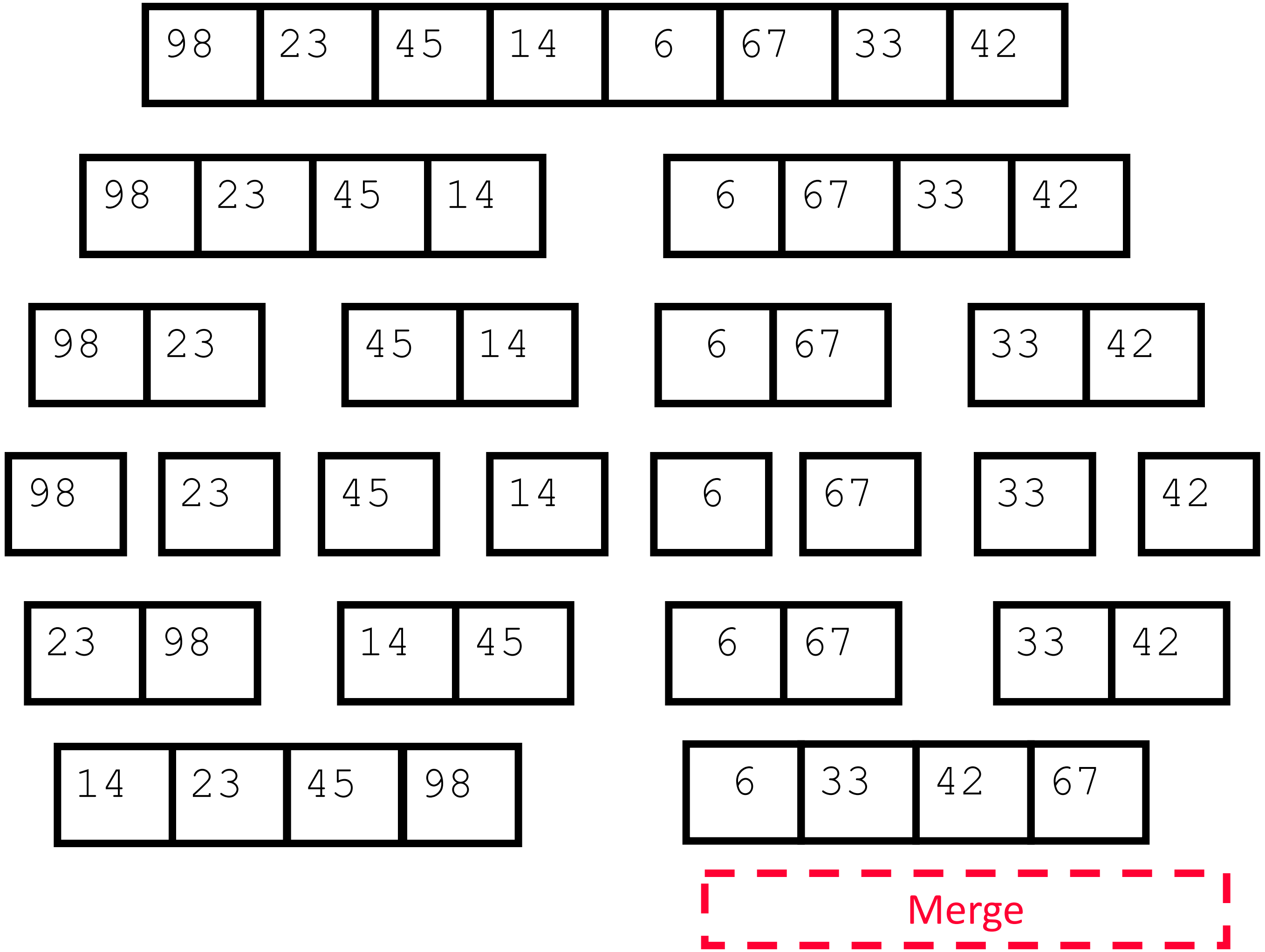

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
Merge

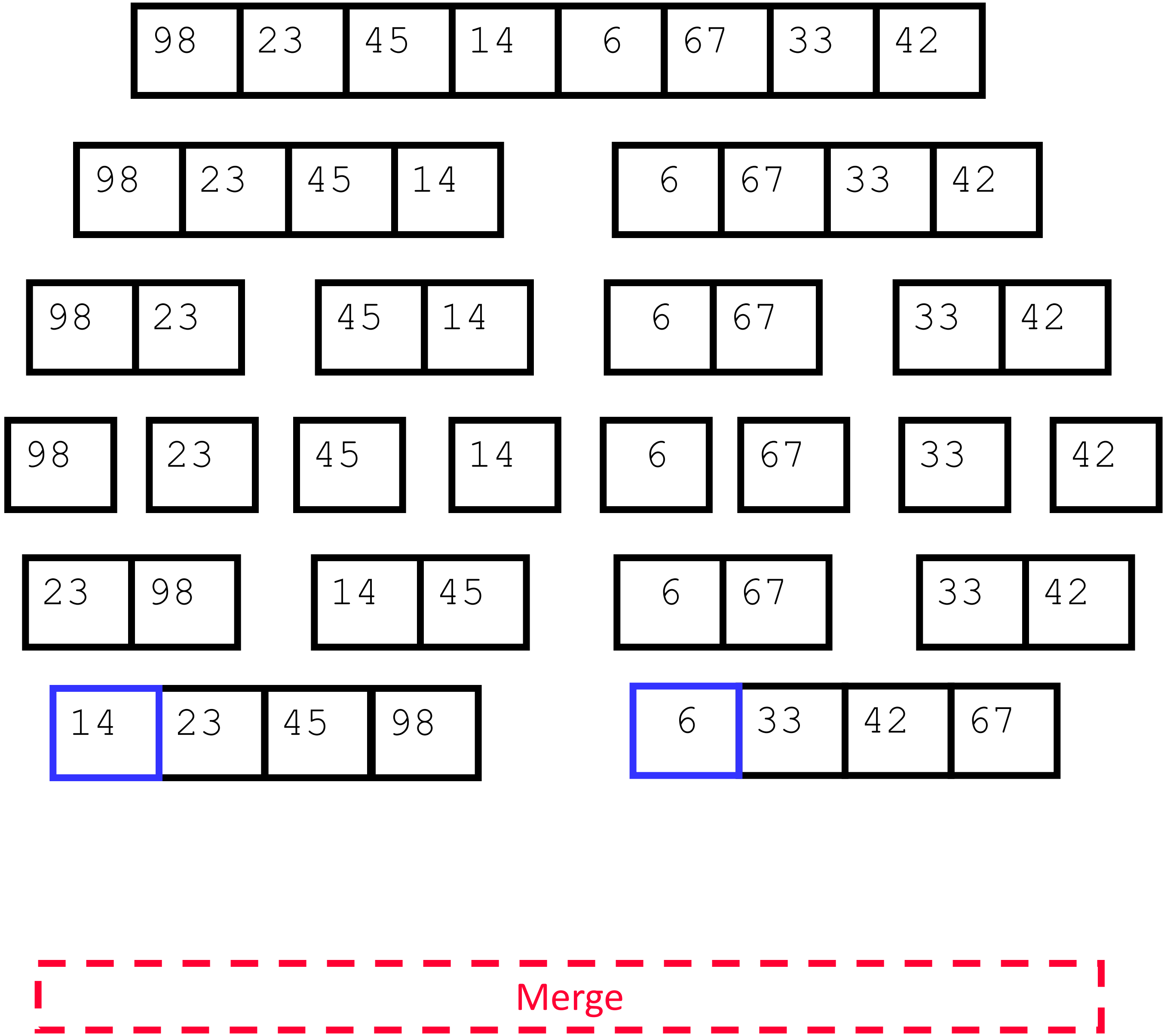


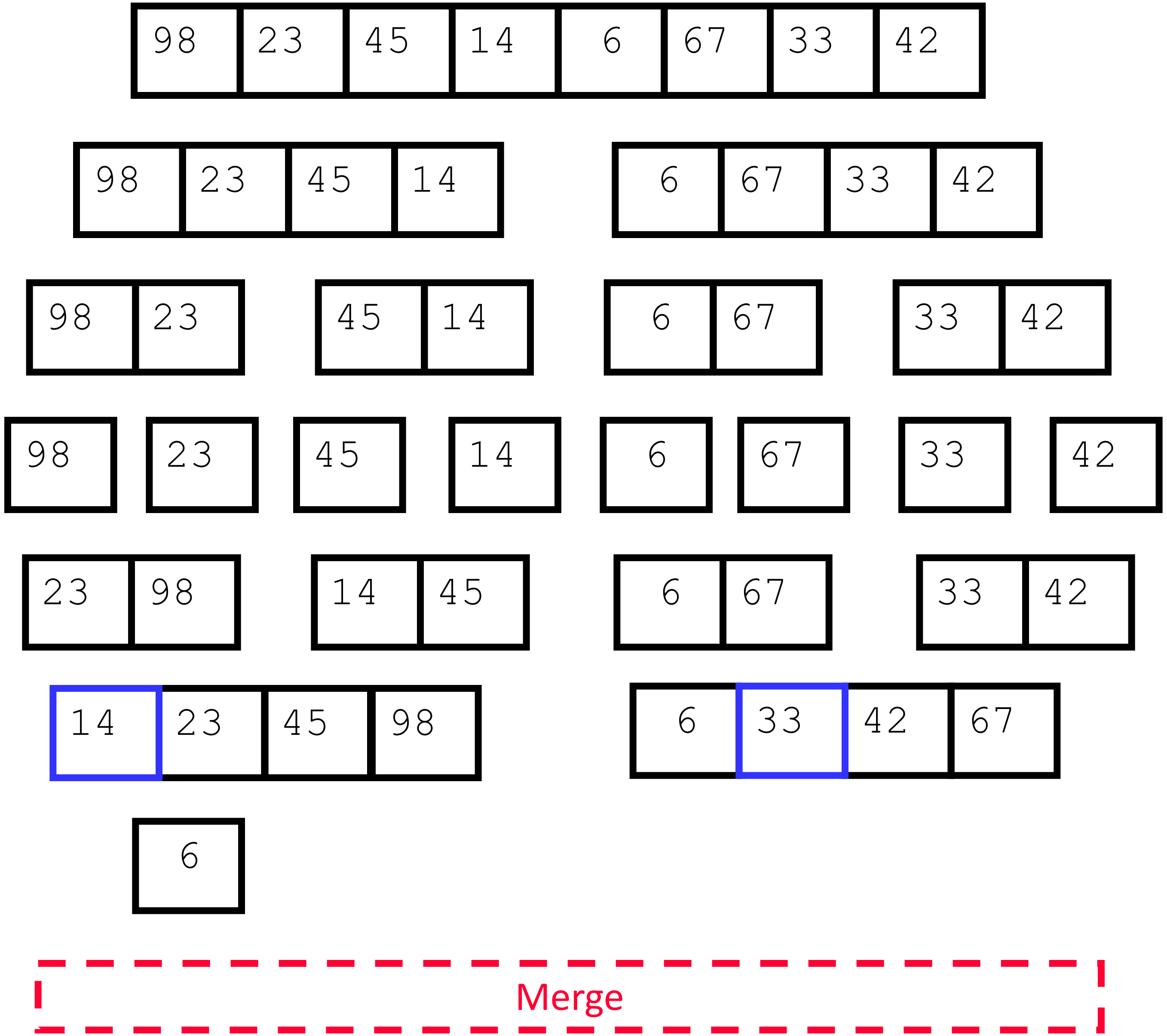
98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6
Merge

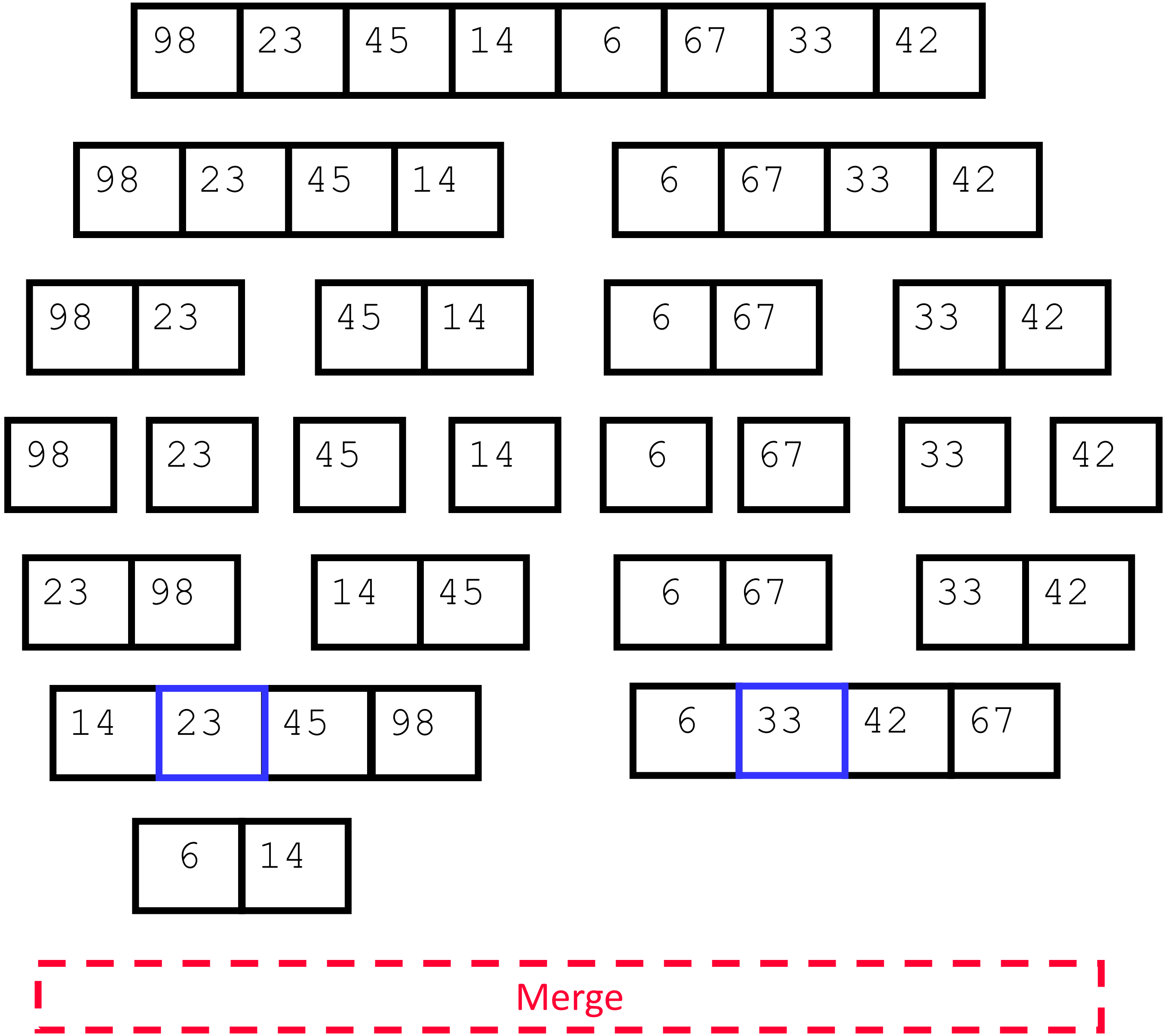

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33
Merge

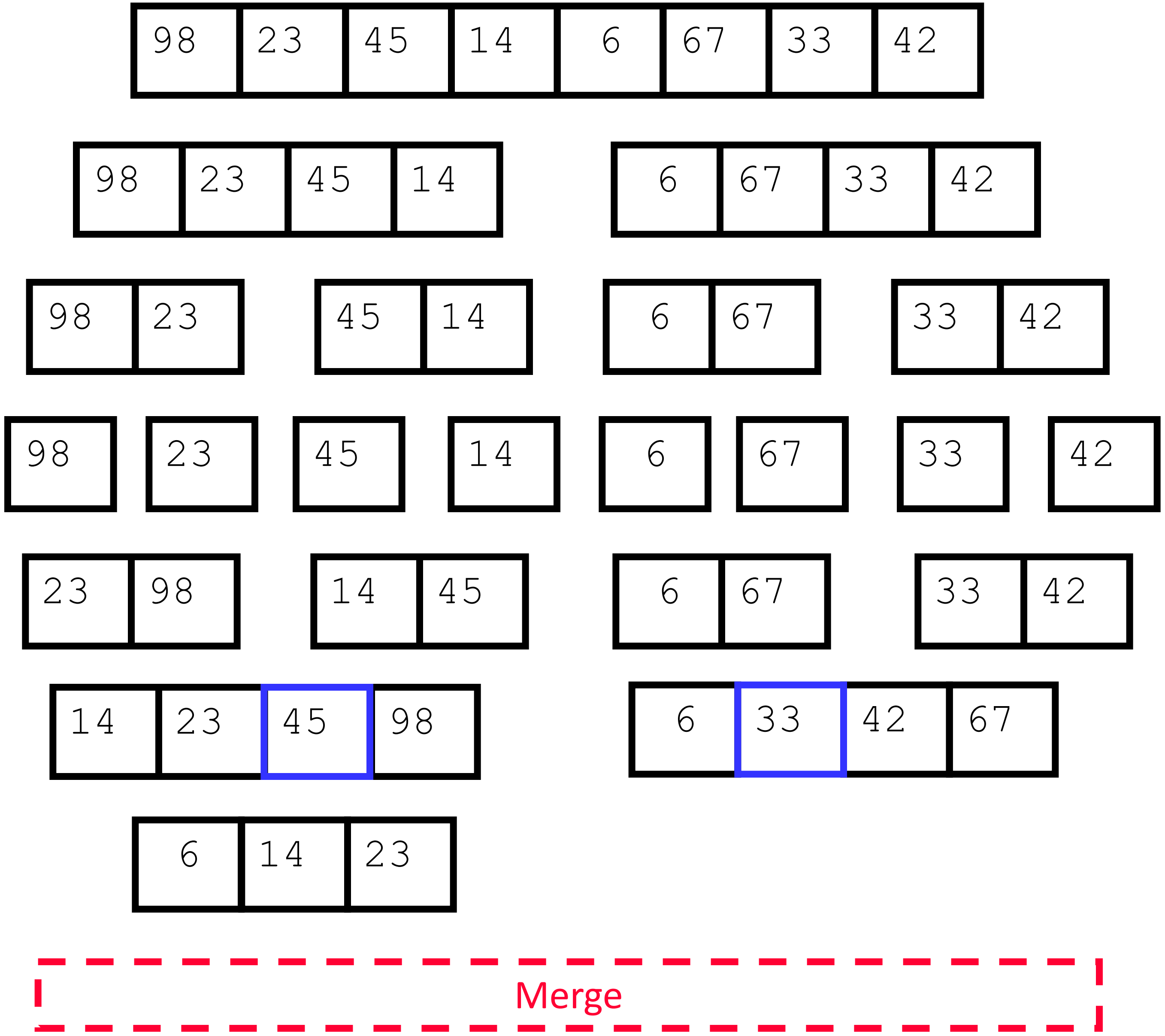

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42
Merge

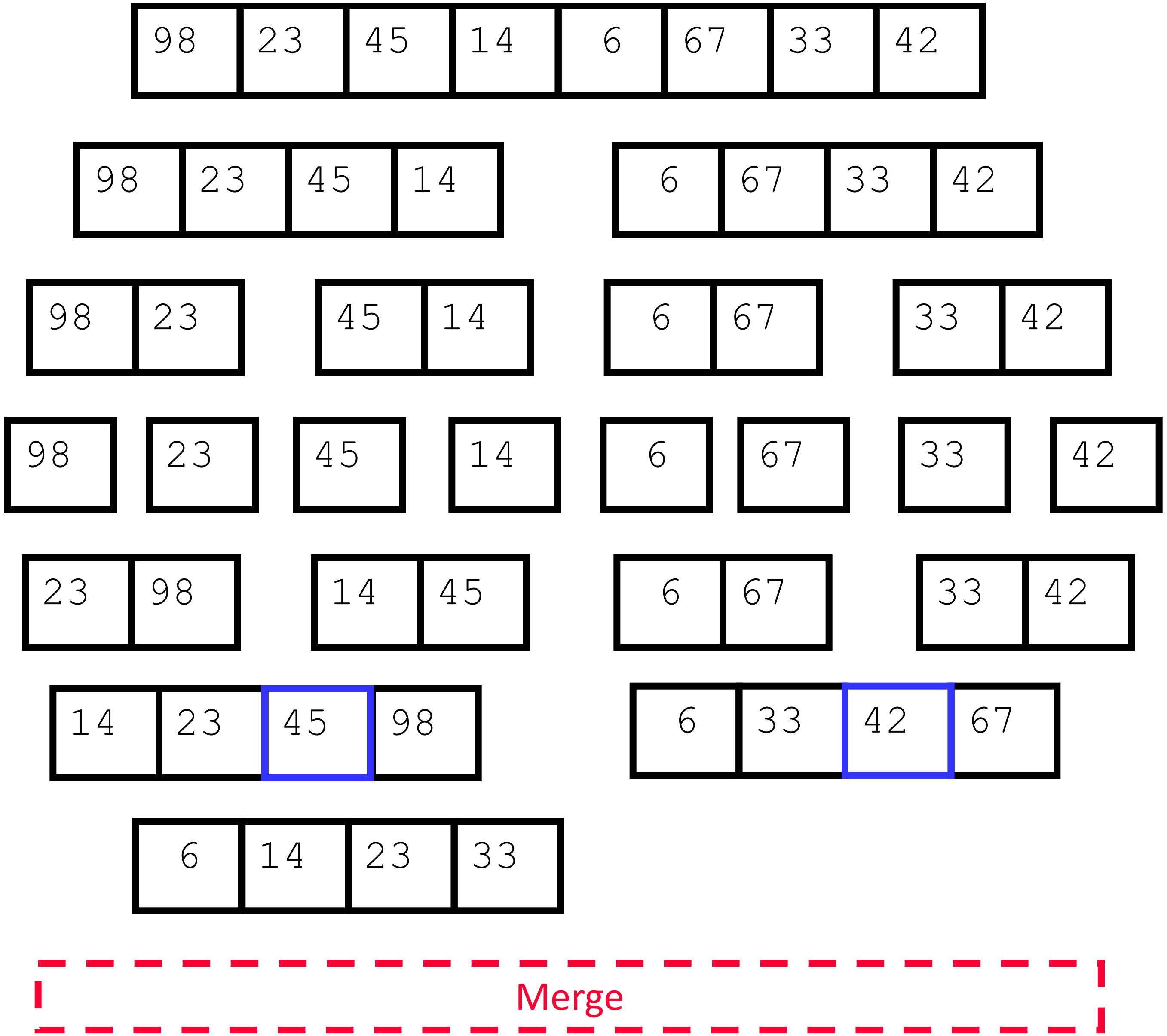

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
Merge

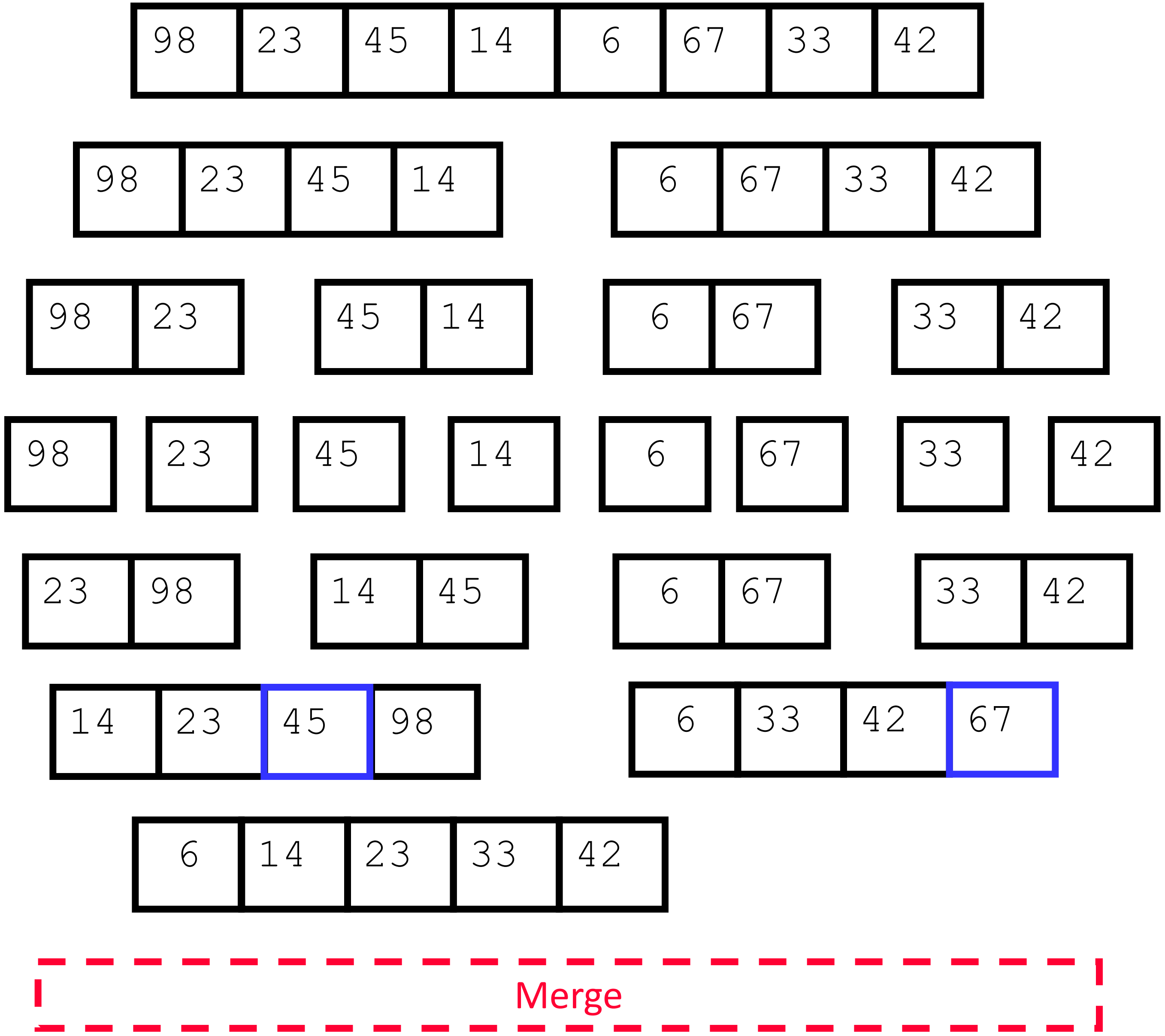

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
Merge

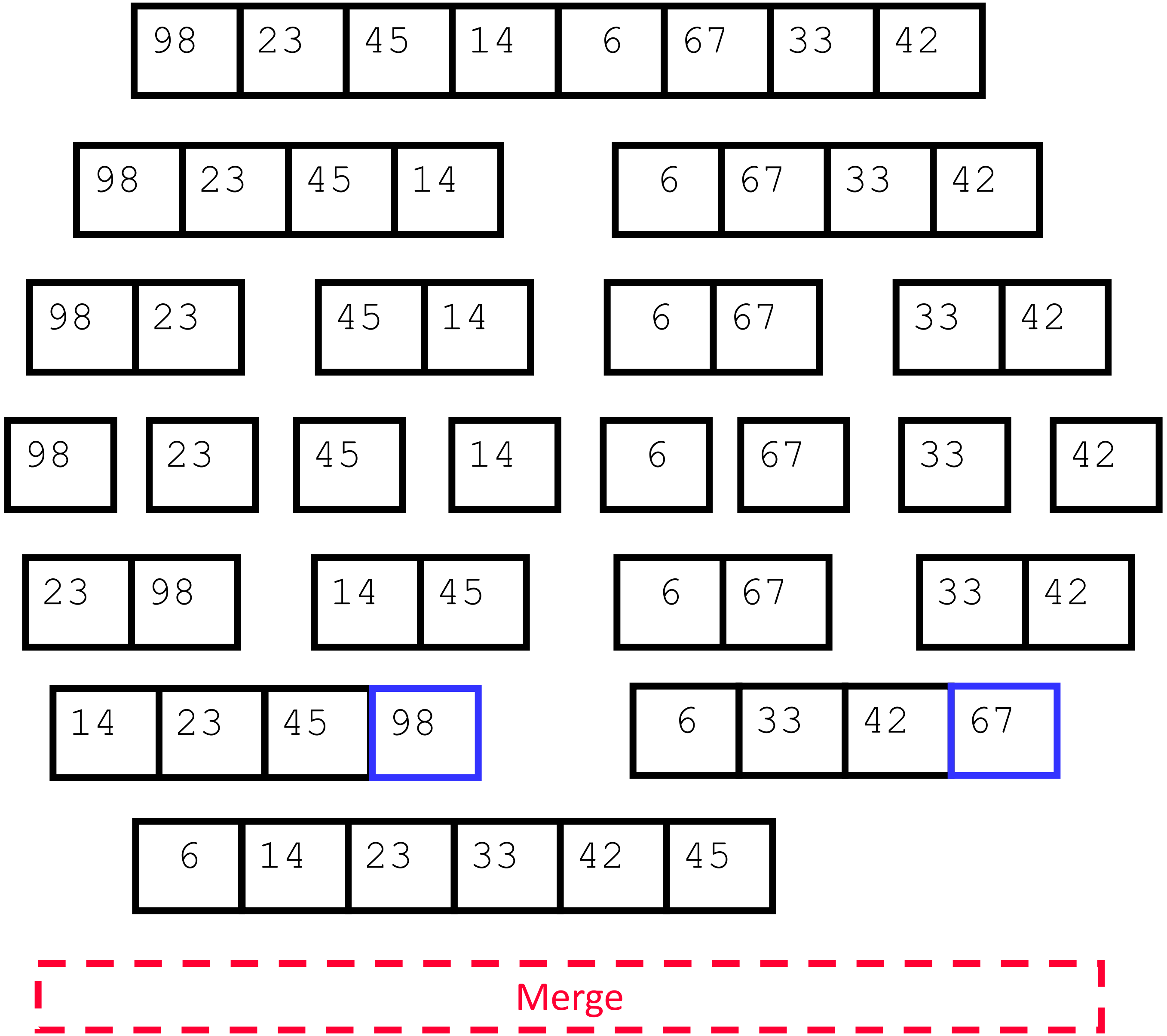

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6
Merge

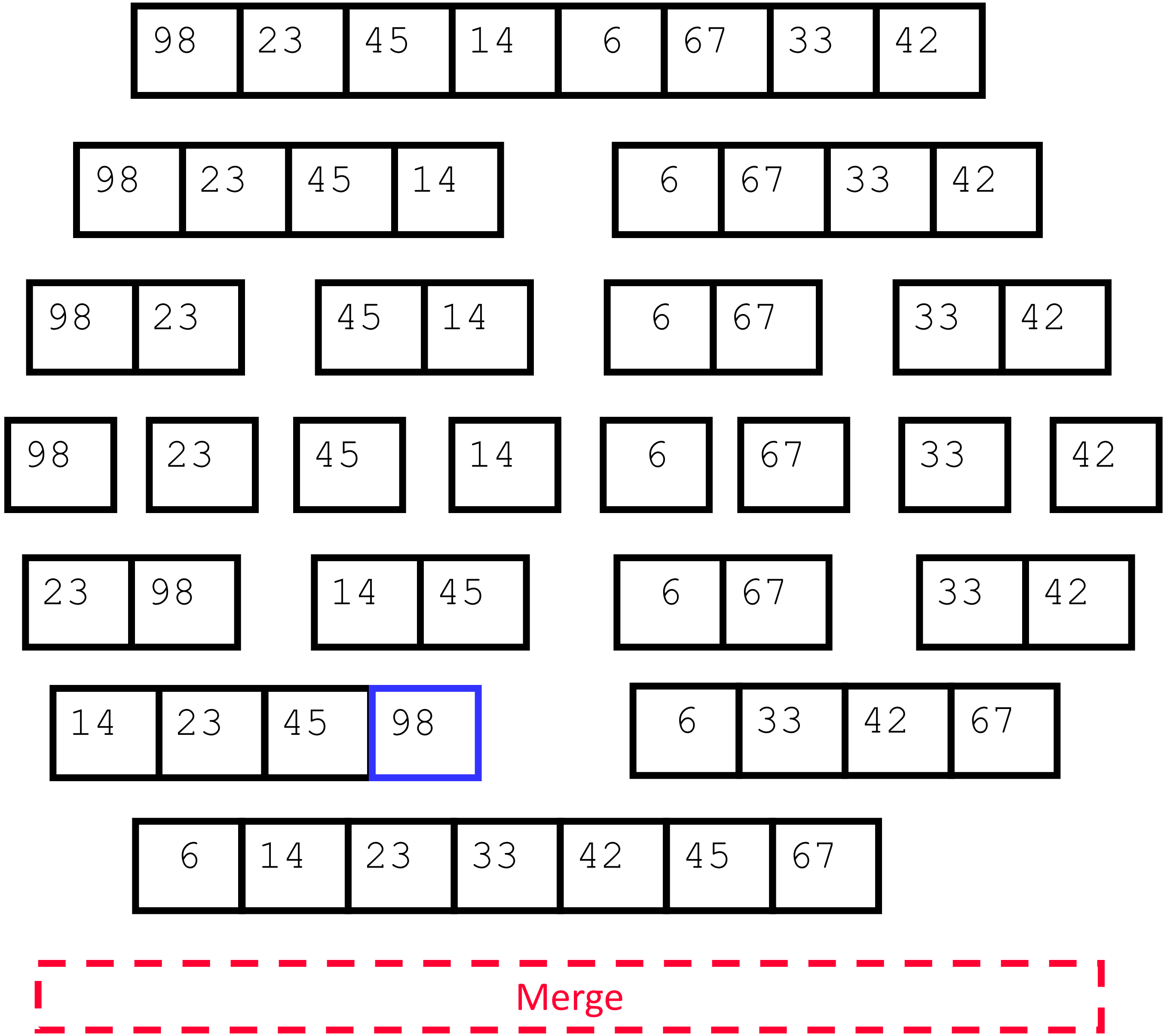

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14
Merge

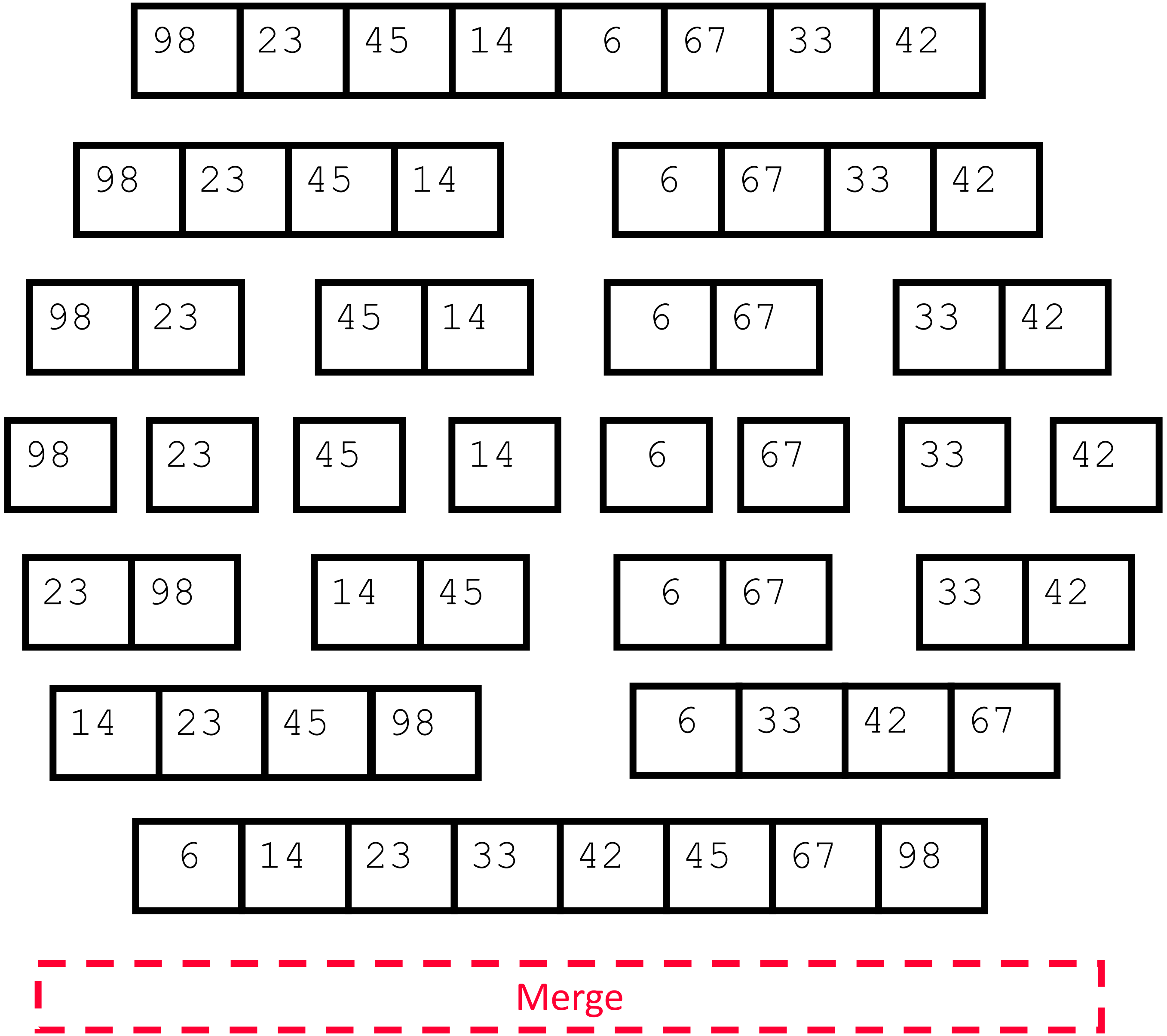

98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67
Merge


98 23 45 14 6 67 33 42
98 23 45 14
6 67 33 42
98 23
45 14
6 67
33 42
98
23
45
14
6
67
33
42
23 98
14 45
6 67
33 42
14 23 45 98
6 33 42 67
6 14 23 33 42 45 67 98
Merge

| 98 | 23 | 45 | 14 | 6 | 67 | 33 | 42 |

| 98 | 23 | 45 | 14 | | 6 | 67 | 33 | 42 |

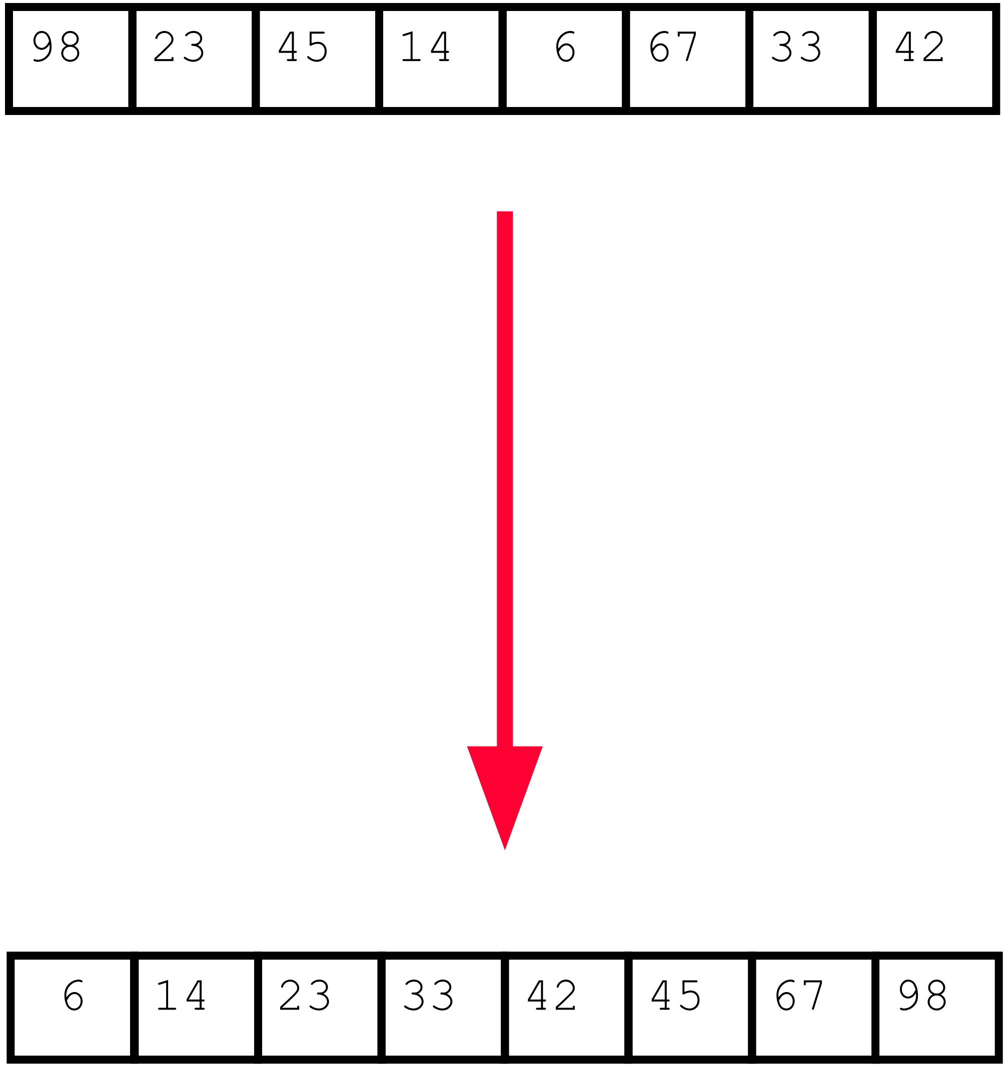
| 98 | 23 | | 45 | 14 | | 6 | 67 | | 33 | 42 |

| 98 | | 23 | | 45 | | 14 | | 6 | | 67 | | 33 | | 42 |

| 23 | 98 | | 14 | 45 | | 6 | 67 | | 33 | 42 |

| 14 | 23 | 45 | 98 | | 6 | 33 | 42 | 67 |

| 6 | 14 | 23 | 33 | 42 | 45 | 67 | 98 |


98 23 45 14 6 67 33 42
6 14 23 33 42 45 67 98

Textbooks & Web References

- Text Book (Chapter 2)
- Reference book i (Chapter 2)
- Reference book ii (Chapter 5)
- www.visualgo.net

Thank you

&

Any question?